АВТОМАТИЗАЦИЯ НАЛАДКИ И ДИАГНОСТИРОВАНИЯ МИКРОУВК

В. А. Гуляев В. И. Кудряшов

АВТОМАТИЗАЦИЯ наладки и диагностирования микроУВК

ЭНЕРГОАТОМИЗДАТ

В. А. Гуляев
В. И. Кудряшов

АВТОМАТИЗАЦИЯ наладки и диагностирования микроУВК

ЭА

МОСКВА

1992

ББК 32.973
Г94
УДК 681.32.004.5

Гуляев В.А., Кудряшов В.И.

Г94 Автоматизация наладки и диагностирования микроУВК. – М.: Энергоатомиздат, 1992. – 256 с.: ил.

ISBN 5-283-01526-2

Рассмотрены диагностические средства поиска неисправностей в различных цифровых устройствах, в том числе и микропроцессорных, на этапах их производства и эксплуатации. Описаны алгоритмы поиска неисправности с помощью различных диагностических устройств (внутрисхемных эмуляторов, логических и сигнатурных анализаторов), а также алгоритм построения тестовых последовательностей и методики их составления.

Для инженеров, занятых разработкой и применением микропроцессорных средств автоматизации и управления.

Г $\frac{2404000000-074}{051(01)-92}$ 227-91

ББК 32.973

Рецензент М.Б. Игнатьев

ISBN 5-283-01526-2

© Авторы, 1992

ПРЕДИСЛОВИЕ

Значительное увеличение производительности труда и улучшение качества изделий в различных отраслях промышленности достигается применением современных микропроцессорных средств вычислительной техники. Сроки создания, внедрения и сопровождения самих микропроцессорных систем (МПС) определяются уровнем технологии их производства и, в частности, эффективностью процессов разработки, отладки и диагностирования. Отладочные и диагностические операции как при разработке, так и при эксплуатации МПС являются весьма трудоемкими, требующими привлечения специалистов высокой квалификации. Сложность микропроцессорных БИС, их специфика видоизменили сами задачи отладки, настройки и диагностирования создаваемых МПС, поскольку необходимо учитывать ограниченное число доступных для проверки входов и выходов, косвенным путем получать информацию о состоянии тех или иных блоков.

В настоящее время сложились определенные методики работы с микропроцессорной техникой и сформировались комплексы технических средств для отладки МПС. Как в составе этих комплексов, так и автономно широко используются логические и сигнатурные анализаторы, которые позволяют обрабатывать диагностическую информацию о динамических, структурных и функциональных свойствах проверяемых устройств в реальном масштабе времени без вмешательства в процесс их функционирования. Они являются незаменимым средством для комплексной отладки аппаратуры и программ МПС.

Дальнейшее совершенствование методов логического и сигнатурного анализа может осуществляться в направлении организации средств поддержки их применения с целью снижения требований к квалификации обслуживающего персонала.

В книге ставится цель изложить "технологию" (в широком смысле) рациональной организации процессов отладки и диагностирования микропроцессорных систем реального времени. Эти вопросы не нашли пока должного отражения в литературе (за исключением [2]), несмотря на их важность и полезность для практических разработок. В доступной читателю литературе содержится описание общих принципов действия микропроцессорных систем и их программного обеспечения, излагаются вопросы выбора архитектуры. Предлагаемая книга в определенной степени восполняет имеющийся пробел и содержит описания принципов, методик и практических приемов использования сигнатурного и логического анализа для организации процедур комплексной отладки и диагностирования МПС при разработке, изготовлении и эксплуатации.

Книга написана на основе оригинальных разработок авторов, в частности на материалах прочитанных ими лекций, и с использованием результатов отечественных специалистов по логическому и сигнатурному анализу, без которых была бы неполной. В связи с этим авторы признательны Г.Х. Новику, П.П. Алексееву, М.И. Королеву за консультации и предоставленные материалы.

Параграф 3.4 написан С.И. Старчихиным, § 7.1 подготовлен О.Е. Ярцевой. Авторы благодарны В.Б. Творогову за помощь в написании § 3.2, Т.А. Якуниной за оформление рукописи и М.Б. Игнатьеву за сделанные при рецензировании замечания, способствовавшие улучшению книги.

Авторы

Глава 1

ОБЩИЕ ВОПРОСЫ ОТЛАДКИ И ДИАГНОСТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

1.1. ОСОБЕННОСТИ МИКРОУВК КАК ОБЪЕКТА НАЛАДКИ И ДИАГНОСТИРОВАНИЯ

Основные понятия и определения. *Неисправность* – это парадоксальная ситуация, которая, с одной стороны, является нежелательной, поскольку приводит к простою и, следовательно, экономическим потерям, а с другой стороны, дает работу сотням тысяч человек, занятых поиском места возникновения неисправности и последующим ремонтом неисправного изделия. Поиск места возникновения неисправности – наиболее сложная и интересная часть работы людей, занятых эксплуатацией и наладкой любых технических устройств. Увеличение сложности техники, ужесточение требований к времени поиска неисправности определили возникновение новой научно-технической дисциплины – диагностики и необходимость роста образовательного ценза обслуживающего персонала.

Техническая диагностика в области вычислительной техники переживает в настоящее время очередной процесс обновления, связанный с изменением элементной базы. До последнего времени развитие элементной базы вычислительной техники шло эволюционным путем.

Наименьшая конструктивная единица ЭВМ – логическая ячейка – реализовывала одну или несколько булевых функций. Логические ячейки вначале выполнялись на базе электронных ламп, затем транзисторов и, наконец, интегральных схем, однако их функциональные возможности изменялись достаточно мало, что позволило использовать одни и те же методы и способы технической диагностики.

В середине 70-х годов в электронной промышленности произошло резкое, можно сказать, революционное изменение технологии, в результате которого стало возможным "упаковывать" в кристалл десятки тысяч различных электрорадиоэлементов (ЭРЭ). Это предопределило появление новых конструктивных элементов – больших интегральных схем (БИС) и сверх-

больших интегральных схем (СБИС). Эти элементы содержат в себе устройства, которые на предыдущей элементной базе занимали целые шкафы, например, постоянное запоминающее устройство емкостью 8 Кбайт (К573РФ4), оперативное запоминающее устройство емкостью 64 Кбайта (К565РУ5), 16-разрядный микропроцессор (то, что в ЭВМ третьего поколения именовалось центральным процессорным устройством) с возможностью непосредственной адресации 1 Мбайта памяти (К1810 ВМ86) и т. д. В этих условиях применение прежних методов диагностики стало практически невозможным.

В настоящее время на базе микропроцессорных комплектов интегральных схем (МКИС) создаются различные средства вычислительной техники – контроллеры для управления сравнительно простыми технологическими процессами, микропроцессорные управляющие вычислительные комплексы (микроУВК) для управления сложными технологическими системами, автоматизации испытаний сложных технических изделий, персональные и профессиональные компьютеры различной мощности и назначения. Все эти устройства отличает возросшая надежность по сравнению с аналогичными изделиями предыдущего поколения, достигнутая за счет увеличения надежности элементной базы, и высокая технологичность, которая позволила резко снизить затраты на производство, что существенно повлияло на их стоимость и дало возможность увеличить количество выпускаемых изделий.

Переход к массовому выпуску средств вычислительной техники различного назначения на базе МКИС ставит задачу повышения качества и правильной организации обслуживания этой техники, поскольку в противном случае в стране не хватит людских резервов для ее обслуживания.

Высокое качество изделия обеспечивается не только соблюдением технологической дисциплины на заводе-изготовителе, но и в первую очередь грамотным проектированием. В понятие грамотно спроектированного изделия входят все компоненты его схемы и конструкции.

Рациональная организация обслуживания микропроцессорных средств вычислительной техники предполагает экономически эффективную организацию их эксплуатации в сочетании с высоким качеством выполнения ремонтных работ и минимальными временными затратами.

Решение задач повышения качества и организации обслуживания невозможно без автоматизации процесса поиска неисправности. В настоящее время разработано большое количество автоматизированных и ручных средств поиска неисправностей в изделиях, выполненных на базе МКИС.

МикроУВК, как и любое другое изделие, может находиться в двух технических состояниях – исправном и неисправном. Состояние микроУВК в данный момент времени определяется соответствием его свойств признакам, установленным в технической документации. МикроУВК считается неисправным, когда хотя бы одно из его свойств не удовлетворяет требованиям, предъявленным в документации. Процесс определения технического состояния именуется техническим диагностированием, которое осуществляется с помощью системы диагностирования. В общем случае (хотя это и не обязательно) целью технического диагностирования является не только определение технического состояния микроУВК, но и поиск дефекта, который вызвал неисправное состояние. Понятия неисправности и дефекта по определениям, приведенным в ГОСТах, достаточно близки, но неисправность может включать в себя и несколько дефектов, каждый из которых заключается в отдельном несоответствии любого из признаков. Поэтому в дальнейшем изложении будут применяться оба термина как синонимы, когда это не будет противоречить их смыслу.

Неисправность может быть вызвана как логической ошибкой разработчика, так и отказом, который по определению заключается в нарушении исправного состояния объекта.

За время своего существования микроУВК проходит следующие этапы: разработку, создание макетного образца, отработку схемы на макетном образце, передачу в производство, отработку схемы по замечаниям, полученным на предыдущем этапе, окончательную передачу изделия в производство, выпуск на заводе-изготовителе и эксплуатацию. На каждом из этапов, начиная с макетного образца, производится определение технического состояния. Для того чтобы разделить сходные процессы поиска неисправностей, на каждом из этапов введен ряд определений:

1. Поиск неисправности (дефекта) – это поиск причины, вызвавшей несоответствие свойств микроУВК или его отдельных частей признакам, записанным в технической документа-

ции. Термин справедлив для любого этапа существования микроУВК.

2. Отладка – процесс поиска неисправностей в изделии или программе на этапе обработки макетного образца изделия или программы.

3. Наладка – процесс поиска неисправностей в изделии во время его подготовки к сдаче после сборки на заводе-изготовителе.

4. Диагностирование – процесс поиска неисправностей в изделии на этапе его эксплуатации.

На каждом из рассмотренных этапов неисправности носят специфичный (именно для данного этапа) характер: отладка связана в основном с выявлением и устранением ошибок разработчика, наладка – с технологическими дефектами, диагностирование – с отказами. В общем случае неисправности, не выявленные на предыдущем этапе, могут проникать на следующий этап существования микроУВК, хотя вероятность их появления тем ниже, чем дальше отстоит данный этап от этапа, на котором эта неисправность не выявлена. Так, на этапе эксплуатации могут выявить неисправности, связанные с ошибкой разработчика, но такие события будут происходить достаточно редко.

С точки зрения поиска неисправности важнейшим параметром неисправности является время ее существования:

$$T_с = T_{об} + T_л, \tag{1.1}$$

где $T_с$ – время существования неисправности; $T_{об}$ – время обнаружения неисправности, т.е. время, которое прошло с момента возникновения неисправности до ее обнаружения; $T_л$ – время локализации источника неисправности.

Выше было сформулировано понятие отказа, однако оно требует некоторой конкретизации с точки зрения временных параметров. Далее под отказом будем понимать нарушение исправного состояния объекта, длительность существования которого превышает время, необходимое на обнаружение неисправности и ее локализацию. Для отказа формула (1.1) имеет вид

$$T_с^{отк} \geqslant T_{об} + T_л, \tag{1.2}$$

где $T_с^{отк}$ – время существования отказа.

В эту категорию входят как постоянно существующие отказы (т.е. такие отказы, которые не могут самовосстанавливаться), так и перемежающиеся отказы-неисправности, которые с течением времени самоустраняются, но длительность которых удовлетворяет выражению (1.2).

Перемежающиеся отказы, время существования которых меньше времени, необходимого на обнаружение и локализацию неисправности, будем именовать сбоями. Для сбоев формула (1.1) примет вид

$$T_с^{сб} < T_{об} + T_л, \qquad (1.3)$$

где $T_с^{сб}$ – время существования сбоя.

Сбои относятся к наиболее трудно локализуемым видам неисправностей. Сбои логично подразделить на два вида: однократные сбои – сбои, которые после восстановления работоспособности больше не возникают, и многократные сбои – сбои, которые возникают с некоторым временны́м интервалом.

Причина возникновения однократных сбоев, как правило, случайна, например: попадание в ИС элементарной частицы, которая вызвала ее ложное срабатывание или изменение логического состояния; возникновение достаточно мощной разовой электрической помехи и т.д. Поиск места возникновения такого рода неисправности невозможен да и не нужен, а программное обеспечение микроУВК должно в случае особо ответственных применений иметь возможность восстановления работоспособности системы без ущерба для объекта управления.

Многократные сбои вызываются самыми различными причинами, в основе которых чаще всего лежит тот или иной дефект. Период возникновения многократных сбоев может быть различным, а промежутки времени между соседними сбоями могут быть не равны. В качестве примера периодического и апериодического сбоев можно рассмотреть ситуацию, в которой неисправность возникает из-за технологического дефекта в ИС, связанного с некачественной пайкой вывода. Предположим, что из-за вибрации в месте пайки возникает неконтакт. Тогда возможны, например, такие причины: из-за биения вала вентилятора в печатной плате возникают колебания, которые будут вызывать сбой с периодом, равным времени одного обо-

рота вала. Тот же эффект может вызываться и вибрацией, вызванной прохождением трамвая по стыку рельса рядом с помещением, в этом случае вибрация будет апериодической и ее период будет зависеть от расписания движения трамваев. Кроме вибрации многократные сбои могут вызываться другими причинами, например в их основе может лежать тепловой эффект.

Поиск места возникновения многократных сбоев тем труднее, чем длительнее период их возникновения. Хороший результат дает способ провоцирования перехода сбоя в разряд постоянных дефектов, что достигается за счет изменения напряжений питания в микроУВК, повышения температуры окружающей среды, простукивания подозрительных мест печатной платы и т.п. После того как удалось добиться стабилизации неисправности, ее локализация производится одним из способов, которые будут описаны ниже.

Анализ особенностей структуры и архитектуры микропроцессорных устройств и комплексов. Микропроцессорные устройства и комплексы разрабатываются на базе микропроцессорных комплектов интегральных схем [13]. В настоящее время существует широкий спектр различных МКИС как универсальных, например: К580, К587, К1810 СССР, I 8080, I 8086 фирмы Intel, M6800 фирмы Motorola, F8 фирмы Fairchild (США) и др., так и специализированных, например: К581, 5536 СССР, CP-1600 фирмы General Instrument, LSI 12/16 фирмы General Automation (США) [14] и др.

Создание средств диагностирования микропроцессорных устройств и комплексов существенно затруднено без анализа основных особенностей их архитектуры.

Вычислительное устройство (рис. 1.1), которое построено на базе МКИС, состоит из четырех групп БИС, образующих: микропроцессор, систему памяти, систему ввода-вывода, систему связи с объектом. Все эти БИС разработаны с учетом совместности интерфейсов, что позволяет организовать шинную структуру вычислительного устройства. При такой организации БИС, входящие в состав вычислительного устройства, непосредственно подключаются к шинам данных, адреса и управления. Характерной чертой интерфейсов микропроцессорных вычислительных устройств является возможность их двунаправленности, т.е. возможность передачи потоков информации в обе стороны по одному проводу.

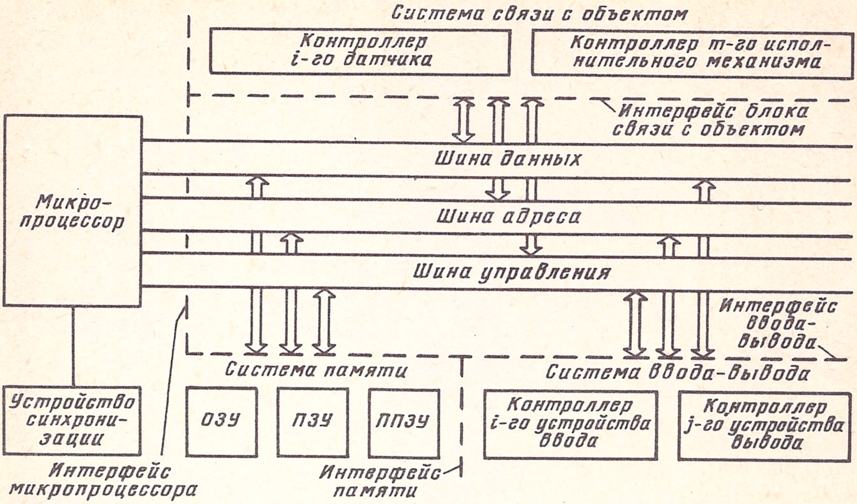


Рис. 1.1. Структурная схема вычислительного устройства на микропроцессорном комплекте интегральных схем

Такая архитектура стала возможной благодаря разработке специальных буферных элементов – шинных формирователей с тремя логическими состояниями, которые как вводятся в состав БИС, так и входят в виде самостоятельных ИС в МКИС. Два логических состояния этих ИС являются стандартными логическими состояниями для используемого типа интегральных схем, т.е. логической 1 (Лог. 1) и логическим 0 (Лог. 0), а третье состояние является высокоимпедансным (оно обозначается как *z*-состояние). Работа шинных формирователей рассматривается на примере шинных формирователей К589АП16. На рис. 1.2, *а* представлен выходной каскад шинного формирователя, работа которого поясняется диаграммой потенциалов (рис. 1.2, *б*). Характерной чертой работы этой схемы является то, что при переходе в *z*-состояние происходит выключение выходных транзисторов *T1* и *T2* и на выходе схемы устанавливается потенциал, равный потенциалу линии. Если все ИС, подключенные к линии, находятся в *z*-состоянии и линия не подключена к уровню Лог. 1, то на ней устанавливается потенциал, равный приблизительно 1,4 В, образующийся за счет подключения к линии входных каскадов шинных формиро-

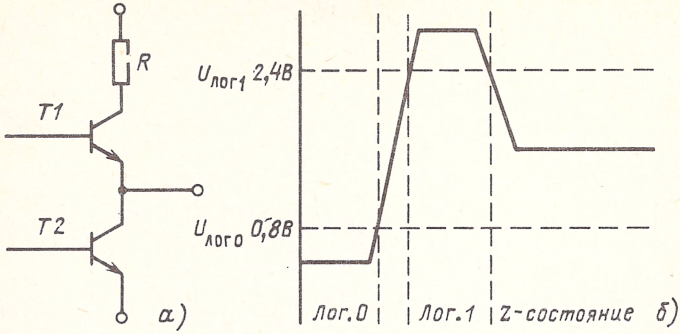


Рис. 1.2. Работа шинного формирователя:

а – упрощенная схема выходного каскада; *б* – диаграмма потенциалов выходного каскада

вателей. В МКИС входят шинные формирователи как инвертирующего, так и неинвертирующего типа. Описанная особенность нового типа интегральных схем – работа с тремя логическими состояниями – является очень существенной с точки зрения диагностики, поскольку в разряд правильно функционирующих схем добавляются схемы, имеющие на своем выходе потенциал, который не соответствует уровням Лог. 0 и Лог. 1, а является неопределенным. При организации системы диагностики устройств, включающих в себя схемы, которые работают с тремя логическими состояниями, возникли следующие вопросы:

правила приема и интерпретации *z*-состояния;

разделение интегральных схем, входящих в состав устройства, на два множества: обычных схем и схем с тремя логическими состояниями (с целью правильного распознавания ошибки).

Микропроцессорные устройства имеют ряд особенностей, связанных с их архитектурой и архитектурой других БИС, на которых выполняются остальные узлы таких устройств. Поэтому успешная разработка систем диагностирования для этих устройств невозможна без тщательного анализа структуры их схемы, временно́й диаграммы работы микропроцессора, временно́й диаграммы всего устройства в целом. Иллюстрация этого утверждения может быть проведена на базе некоторого, специально переусложненного, гипотетического микропроцессорного устройства, использующего микропроцессорный на-

бор К580. Структурная схема этого устройства изображена на рис. 1.3. Здесь специально выбрана структура, позволяющая осуществлять обращение к внутренним ресурсам устройства не только со стороны собственного микропроцессора, но другими процессорами со стороны межмашинной магистрали. Такая архитектура изображенного микропроцессорного устройства показывает, что при разработке систем диагностирования необходимо учитывать не только работу микропроцессора этого устройства со своими ресурсами, но и возможность их использования другими процессорами, за счет чего в совместно используемых схемах возникают наложения обращений от разных процессоров и время выполнения операций собственным процессором устройства увеличивается.

На рис. 1.4 изображена обобщенная временная диаграмма работы микропроцессора К580ИК80А и схемы К580ВК28. Анализ этой временной диаграммы показывает, что:

в процессе выполнения любой команды шины данных и адреса микропроцессора имеют в машинных тактах Т3, Т4 и Т5 неопределенное и плавающее состояние;

синхронизация работы осциллографа, логического анализатора, сигнатурного анализатора может производиться по фронтам сигналов С, П, $\overline{\text{В}}$, а также по фронтам сигналов команд: ЧТЕН, ВВ, ППР, ЗАП, ВЫВ;

использование в качестве синхронизирующего импульса сигналов Ф1 и Ф2 в сигнатурных анализаторах невозможно, поскольку их использование приводит к неправильному приему данных в сигнатурный анализатор с шин данных и адреса.

Работа схемы постоянного запоминающего устройства (ПЗУ) особых сложностей не имеет, и оно диагностируется достаточно просто.

Оперативные запоминающие устройства современных микропроцессорных вычислительных устройств и комплексов в настоящее время строятся с использованием двух типов схем: статических и динамических ОЗУ.

Поскольку статическое ОЗУ можно представить в виде набора адресуемых триггерных регистров, то его диагностирование не должно вызвать особых трудностей.

Динамическое ОЗУ – наиболее распространенный из-за его большой емкости тип оперативной памяти. Эти БИС имеют тот недостаток, что информация в памяти должна периодически восстанавливаться (регенерироваться). Для этого в устройство

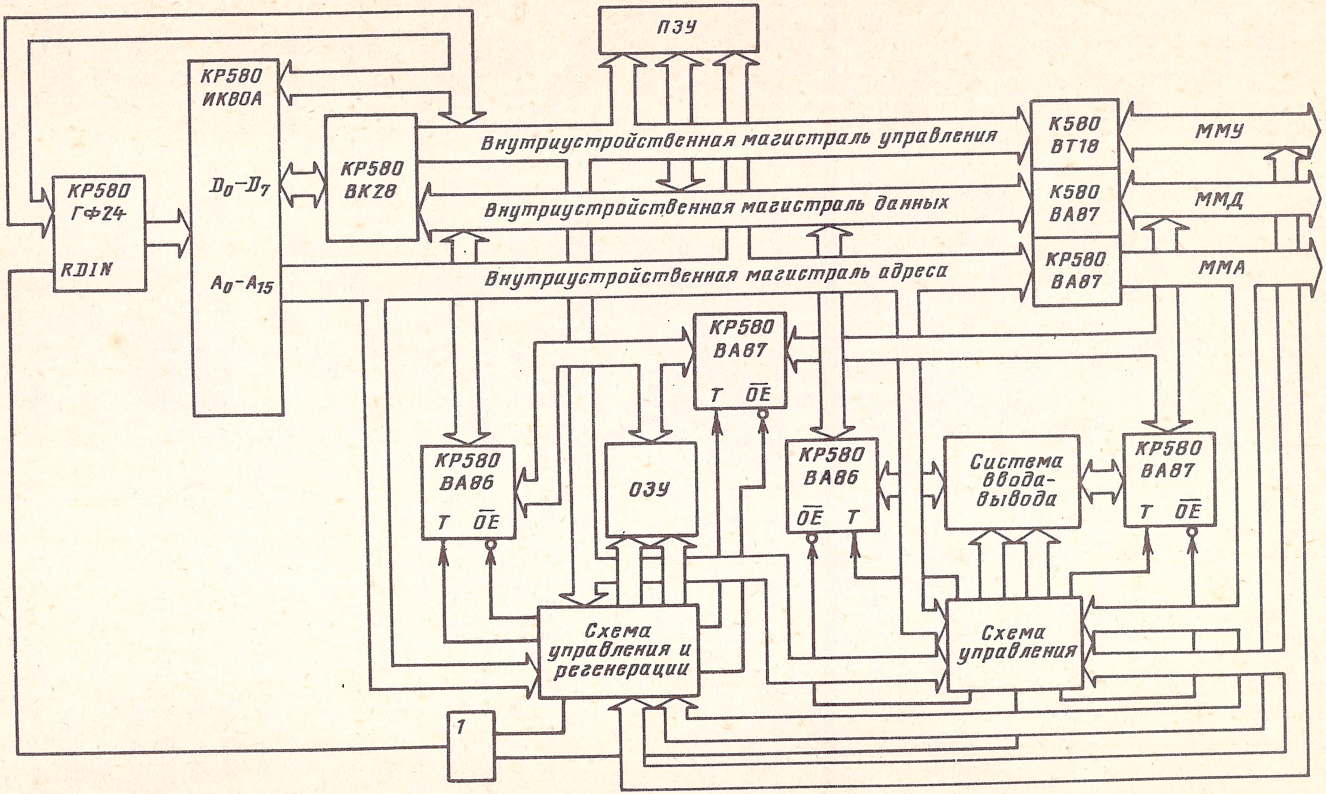


Рис. 1.3. Структурная схема микропроцессорного устройства

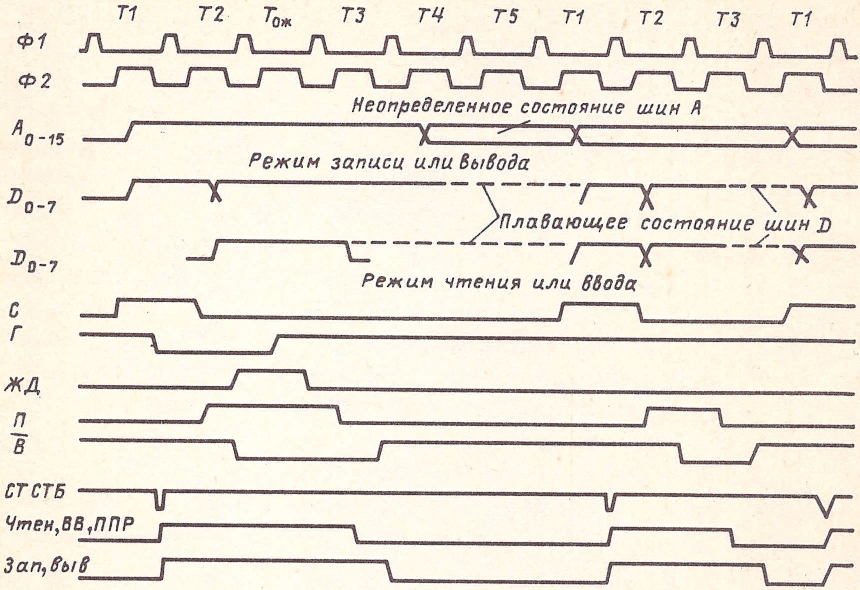


Рис. 1.4. Обобщенная временная диаграмма работы БИС КР580ИК80А и КР580ВК28

памяти вводят специальную схему, обеспечивающую регенерацию информации. Возможны три типа регенерации:

скрытая регенерация, при которой восстановление информации происходит во время подготовки микропроцессора к обращению к памяти (такт *T1* и начало такта *T2* на рис. 1.4), после окончания обращения (конец такта *T3*, такты *T4* и *T5*) или при обращении микропроцессора к другим устройствам; при этом количество обращений к ОЗУ с регенерацией должно быть не меньше указанного в документации на данную БИС ОЗУ за указанный там же промежуток времени;

распределенная регенерация, при которой весь требуемый для нее интервал времени разбивается на равные промежутки, в течение которых производится одно обращение к ОЗУ с регенерацией; количество обращений за данный интервал должно быть равно оговоренному количеству для данной схемы, указанному в ее ТУ;

пакетная регенерация, которая выполняется следующим образом: один раз за оговоренный для данной БИС ОЗУ интер-

вал времени проводится восстановление информации во всех ячейках памяти, для чего последовательно формируется требуемое количество обращений к ОЗУ с регенерацией.

В ОЗУ микропроцессорных устройств может применяться любой тип регенерации, возможно использование их комбинаций или всех трех типов регенерации одновременно. Каждый из этих типов имеет свои достоинства и недостатки. Общим недостатком двух последних типов является обязательное уменьшение скорости работы микропроцессорного устройства за счет наложения обращений к ОЗУ на циклы регенерации. Первый же тип регенерации может не обеспечить необходимого количества циклов регенерации в каждый интервал времени в зависимости от выполняемой программы, что может привести к потере информации.

Другой особенностью динамических ОЗУ является мультиплицированная шина адреса, на которую во время обращения поочередно подается младшая и старшая половины шины адреса микропроцессора.

Перечисленные выше особенности динамических ОЗУ должны учитываться при разработке систем диагностирования. Очевидно, что для обеспечения стабильной "картинки" на экране осциллографа, а также стабильной сигнатуры на сигнатурном анализаторе необходимо иметь возможность отключать на время поиска неисправности схему регенерации.

Сама схема регенерации наиболее просто проверяется в автономном режиме работы, хотя это не всегда возможно обеспечить.

Для проверки правильности работы коммутатора адреса ОЗУ необходимо тщательно, с учетом характеристик вашего сигнатурного анализатора подбирать синхронизирующие сигналы.

БИС ввода-вывода имеют особенности при диагностировании. Но наиболее сложными в этом плане являются контроллер последовательного интерфейса и таймер (КР580ВВ51 и КР580ВИ53 соответственно). Эти БИС могут работать с асинхронными сигналами. Поиск неисправностей для них достаточно затруднен, а применение метода сигнатурного анализа для асинхронных схем не позволяет полностью диагностировать эти схемы.

Еще одной особенностью архитектуры микропроцессорных устройств, как видно из табл. 1.1, является практически пол-

Таблица 1.1

Обозначение микропроцессорного комплекта ИС	Тип микропроцессорного комплекта ИС	Наличие встроенных средств контроля и диагностики	Возможность использования шинной организации при построении устройств на базе данного комплекта
К586	Специализированный	Есть	Есть
К580	Универсальный	Нет	,,
К581	Специализированный	,,	,,
К584	Универсальный	,,	Однонаправленные шины
К586	Специализированный	,,	Есть
К587, КР587	Универсальный	,,	,,
К588	,,	,,	,,
К589	,,	,,	,,
83–К1883	,,	,,	,,
К1810	,,	,,	,,

ное отсутствие встроенных средств контроля и диагностики. Основными причинами этого являются: большое количество функций, реализованных в каждой БИС, ограниченное количество элементов, которые могут быть размещены на одном кристалле, и ограниченное количество выводов в корпусе БИС, что не позволило разместить в ней встроенные средства контроля и диагностики. Отсутствие встроенных схем контроля и диагностики существенно затрудняет организацию системы диагностики, поскольку невозможно не только автоматически производить поиск неисправности, но и определить правильность функционирования микропроцессорного устройства или комплекса во время выполнения рабочих программ.

Рассмотрим вопрос возможности организации в микропроцессорном устройстве самодиагностирующей системы. Для этого представим микропроцессорное вычислительное устройство на рис. 1.1 в виде структурного графа $G(E, U)$ (рис. 1.5), в котором вершины $e_i \in E$ представляют собой различные системы вычислительного устройства, а другие $u_{ij} \in U$ – связи между ними. Для более точного отображения функций интерфейсов в

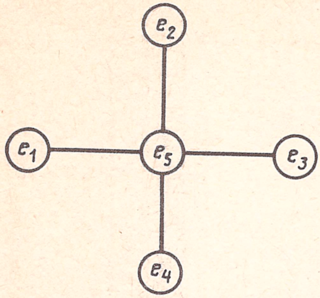


Рис. 1.5. Структурный граф вычислительного устройства, изображенного на рис. 1.1

граф $G(E, U)$ введем дополнительную вершину e_5, которая представляет собой совокупность интерфейсных схем всех систем устройства.

Рассмотрим полученную графовую модель с точки зрения концепции "тест–элемент". В [15–17] показано, что вычислительная система S обладает возможностью самодиагностики, если ее можно разделить на n взаимоисключающих дискретных подсистем S_i, $i = \overline{1,n}$, таких, что для них выполняются следующие условия:

1. Подсистемы S_i, $i = \overline{1,n}$ должны быть диагностируемы, т.е. для каждой подсистемы S_i должна быть известна серия входных воздействий и эталонная реакция на них.

2. Среди подсистем S_i, $i = \overline{1,n}$ существуют диагностирующие подсистемы S_{id}, способные выполнять следующие функции:

f_1 – подача на вход проверяемой системы контролирующего набора;

f_2 – анализ реакции подсистем путем сравнения ее с эталонной;

f_3 – переход в зависимости от результата анализа предыдущей операции;

f_4 – останов с индикацией места отказа.

3. Подсистемы S_i и S_{id} должны быть связаны между собой.

4. Должна быть гарантирована работоспособность хотя бы одной диагностирующей подсистемы.

Разобьем граф $G(E, U)$ на подсистемы S_i, каждой из которых будет соответствовать одна из вершин e_i графа $G(E, U)$:

подсистеме микропроцессора	$S_1 - e_1$;
подсистеме памяти	$S_2 - e_2$;
подсистеме ввода-вывода	$S_3 - e_3$;
подсистеме связи с объектом	$S_4 - e_4$;
подсистеме интерфейса	$S_5 - e_5$.

Таблица 1.2

№ подсистемы	Обозначение подсистемы S_i	Выполнение условия 1	Выполнение условия 2				Выполнение условия 3	Должна ли система иметь гарантированную работоспособность? Условие 4
			f_1	f_2	f_3	f_4		
1	S_1	+	+5	+2,5	+	+3,5	–	+
2	S_2	+	–	+1,5	–	–	–	+
3	S_3	+	–	–	–	+1,5	–	+
4	S_4	+	–	–	–	–	+	–
5	S_5	+	+1	+1,2	–	+1,3	–	+

Для анализа выполнения условий 1–4 всеми системами составим табл. 1.2, в которой выполнение условия одной подсистемой будет отображаться знаком плюс, невыполнение условия – знаком минус, выполнение условия совместно с одной из других подсистем – знаком плюс и номером подсистемы, совместно с которой это условие выполняется.

На основании табл. 1.2 можно определить диагностирующую подсистему S_{id}. Поскольку ни одна из подсистем S_i не выполняет полностью условия 2, то диагностирующей подсистемой должна стать подсистема S_{id}, которую можно определить как объединение подсистем S_k, полностью выполняющих условие 2:

$$S_d = \bigcup_{k=1}^{i} S_k, \tag{1.4}$$

где

$$S_k \subset S_i.$$

Из (1.4) и табл. 1.2 следует, что

$$S_d = S_1 \cup S_2 \cup S_3 \cup S_5. \tag{1.5}$$

Таким образом, как следует из (1.5), в диагностирующую подсистему S_d должны войти подсистемы микропроцессора, памяти, ввода-вывода, интерфейса, т.е. практически все оборудование микропроцессорного устройства. Более того, чтобы микропроцессорное устройство было самодиагностирующимся, согласно условию 4 подсистемы, входящие в S_d, должны иметь гарантированную работоспособность.

Проведенный анализ позволяет сделать следующие выводы: микропроцессорное вычислительное устройство (см. рис. 1.1) не является самодиагностирующимся устройством, поскольку диагностирующая подсистема включает в себя большинство оборудования устройства; шинная организация и отсутствие встроенных средств контроля и диагностики не позволяют осуществить эффективный программно-аппаратный контроль за работой устройства, т.е. осуществлять предварительную локализацию неисправности с точностью хотя бы до подсистемы или группы подсистем.

Отсюда следует, что для поиска неисправности в микропроцессорных вычислительных устройствах типа, представленного на рис. 1.1, необходимо использовать специализированное внешнее диагностическое оборудование.

Внешние диагностические устройства по методам стимуляции проверяемого цифрового устройства и оценки его реализации могут быть разделены на следующие группы:

1) функциональные тестеры, осуществляющие стимуляцию проверяемых схем и анализ их реакции через краевые разъемы платы, на которой они расположены;

2) устройства поэлементного контроля, осуществляющие одновременный контакт со всеми выводами одной (или всех расположенных на печатной плате) интегральной схемы, входящей в состав проверяемого устройства. При этом правильность функционирования каждой интегральной схемы определяется индивидуально (проверяемая интегральная схема "изолируется" от остальных схем, входящих в состав устройства). Устройства поэлементного контроля могут быть как групповыми, осуществляющими контакт со всеми элементами проверяемого устройства, так и индивидуальными, подсоединяемыми к одной микросхеме;

3) внутрисхемные эмуляторы, осуществляющие стимуляцию проверяемых схем и анализ их реакции через разъем микро-

процессора проверяемого устройства. Микропроцессор при проверке вынимается и проверяется отдельно;

4) тестеры, осуществляющие только регистрацию результатов стимулирующего воздействия, организованного тем или иным способом внутри проверяемого устройства: логические анализаторы, сигнатурные анализаторы, регистраторы переходных состояний, логические зонды и т.д. Эти приборы производят последовательный съем реакции с контактов проверяемого устройства, а анализ реакции чаще всего производит человек, осуществляющий поиск неисправности;

5) комбинированные системы, в которых объединены два или более различных способа диагностирования.

Влияние состава микропроцессорного комплекса на систему диагностирования. Современные МПК представляют собой мощные микропроцессорные вычислительные устройства, которые состоят из одного или нескольких центральных процессорных устройств (ЦПУ) и набора устройств расширения функций (устройств расширения памяти, арифметического расширителя, устройств связи с ЗУ на магнитных дисках, устройств цифрового и аналогового ввода-вывода и т.д.). Устройства, входящие в состав МПК, представляют собой отдельные печатные платы, которые вставляются в блок микропроцессорного комплекса, объединяющий их с помощью межмашинной магистрали (рис. 1.6).

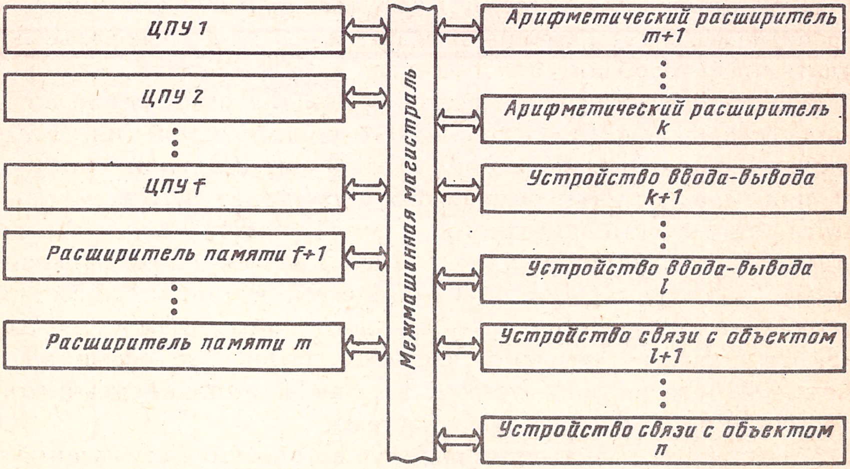


Рис. 1.6. Обобщенная структурная схема МПК

Проведем анализ диагностируемости МПК по методике, использованной выше. Предположим, что состав устройств, входящих в МПК, аналогичен составу устройств МПК (рис. 1.7). Представим МПК (см. рис. 1.6) в виде графа $G^1(E^1, U^1)$ (рис. 1.8), вершины которого $e_i \in E^1$, $i = \overline{1,n}$ представляют собой различные устройства, входящие в состав МПК, а дуги $u^1_{ij} \in U^1$ – связи между ними. Для более точного отображения функций интерфейсов введем в граф $G(E^1, U^1)$ дополнительные вершины q_i, $i = \overline{1,n}$, представляющие собой интерфейсные схемы каждого из устройств, связывающие их с ММ, и вершину q^1_{n+1}, представляющую собой интерфейсную схему ММ.

Разобьем систему S^1, представляющую собой все оборудование МПК, на подсистемы S^1_i, $i = \overline{1,p}$, которые представляют собой объединения однотипных устройств, входящих в МПК, или отдельные устройства:

$S^1_1 = e_1$ – ЦПУ;

$S^1_2 = \bigcup_{i=2}^{f} e^1_i$ – множество ЦПУ;

$S^1_3 = \bigcup_{j=f+1}^{m} e^1_j$ – множество устройств расширения памяти;

$S^1_4 = \bigcup_{t=1}^{n+1} q^1_t$ – множество интерфейсных схем;

$S^1_5 = \bigcup_{h=m+1}^{k} q^1_h$ – множество арифметических расширителей;

$S^1_6 = \bigcup_{g=k+1}^{l} e_g$ – множество устройств ввода-вывода;

$S^1_7 = \bigcup_{r=l+1}^{n} e_r$ – множество устройств связи с объектом.

Произведем анализ полученной графовой модели МПК с точки зрения концепции "тест–элемент" по табл. 1.3.

Анализ табл. 1.3 показывает, что множество подсистем не содержит ни одной подсистемы, полностью выполняющей

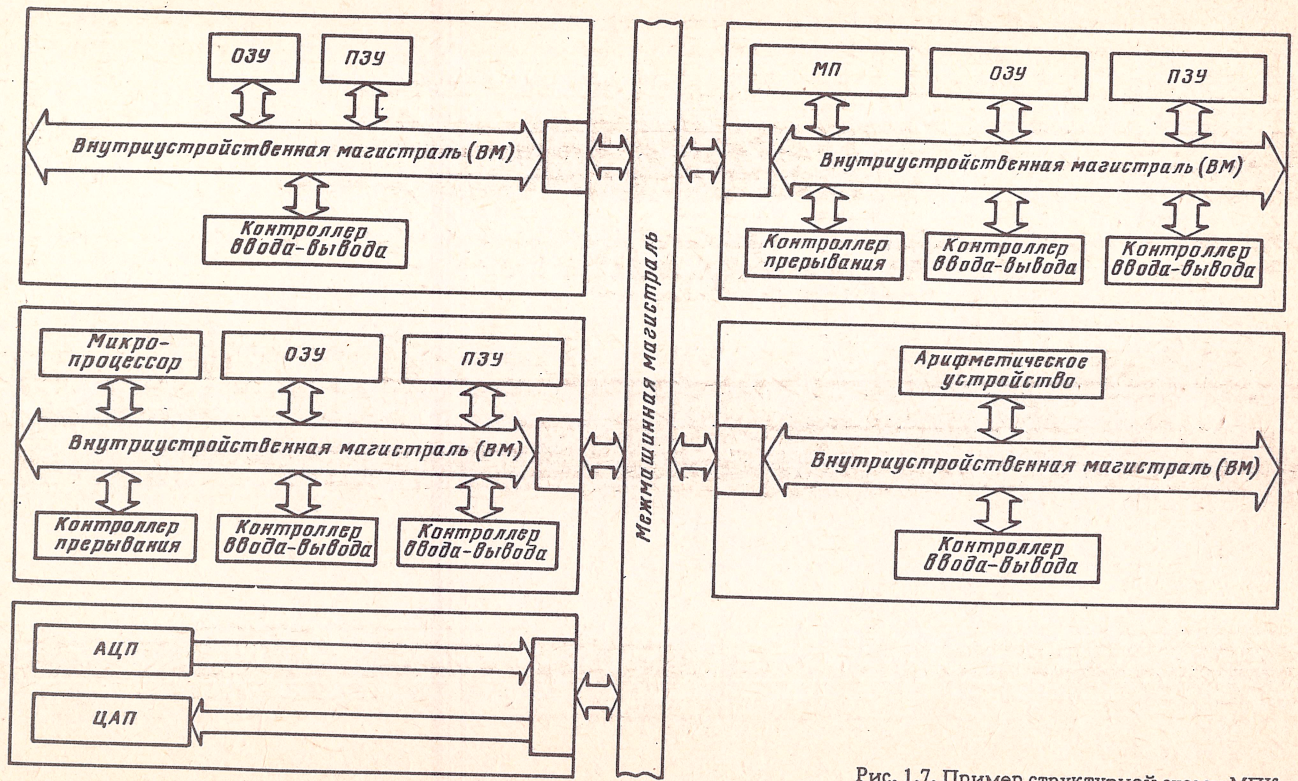


Рис. 1.7. Пример структурной схемы МПК

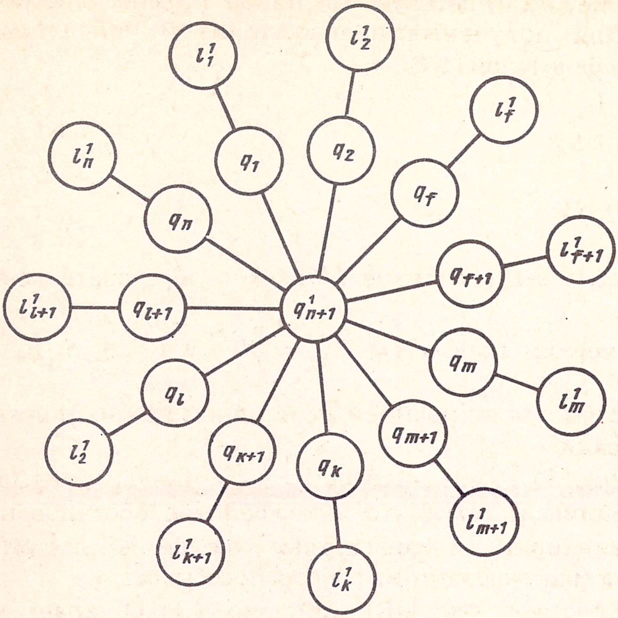


Рис. 1.8. Графовая модель МПК

Таблица 1.3

№ подсистемы	Обозначение подсистемы S_i	Выполнение условия 1	Выполнение условия 2				Выполнение условия 3	Необходимость гарантированной работоспособности подсистемы
			f_1	f_2	f_3	f_4		
1	S_1	+	+4	+	+	+	+	+4
2	S_2	+	+4	+	+	+	+	+4
3	S_3	+	–	+	–	–	+	–
4	S_4	+	+	–	–	–	+	+(1 V 2)
5	S_5	+	–	–	–	–	+	–
6	S_6	+	+	–	–	–	+	–
7	S_7	+	+	–	–	–	+	–

условие 2, т.е. не существует ни одной диагностической подсистемы. Для получения диагностической подсистемы воспользуемся формулой (1.4):

$$S^1_{1d} = S^1{}_1 \cup S^1_4; \tag{1.6}$$

$$S^1_{2id} = e^1_i \cup S^1_4. \tag{1.7}$$

Таким образом, в системе S^1 можно образовать множество диагностических подсистем $S^1_d = S^1{}_{1d} \cup (\bigcup_{i=2}^{f} S_{2id})$, причем работоспособность по крайней мере одной из них должна быть гарантирована.

Включение S^1_4 во все диагностические подсистемы множества S^1_d приводит к тому, что интерфейсное оборудование для связи с межмашинной магистралью каждой из плат МПК должно иметь гарантированную работоспособность.

Отсюда следует, что МПК (см. рис. 1.6) обладает возможностью самодиагностики при условии гарантированной работоспособности интерфейсного оборудования всех плат, входящих в состав МПК, и гарантированной работоспособности одного из центральных процессорных устройств, которое должно при этом выполнять условия $f_1 - f_4$. Это означает, что каждая из центральных процессорных плат должна иметь в своей памяти набор диагностических тестов, иметь возможность связи с устройством отображения (пультом управления, дисплеем, телетайпом и т.д.). При этом в связи с тем, что согласно (1.2) центральная процессорная плата не является самодиагностирующейся системой, а также с тем, что внутренние ресурсы центральной процессорной платы обычно являются недоступными со стороны межмашинной магистрали, автоматический поиск неисправности в центральном процессорном устройстве чрезвычайно затруднен. В [37] приводится система автоматического тестирования МПК, состоящего из ЦПУ, устройств расширения оперативной и постоянной памяти, устройств последовательного и параллельного ввода-вывода. Разрешающая способность такой тестовой диагностической системы невелика из-за шинной организации МПК, плат и невозможности локализации неисправности программным

путем в таких схемах, как схемы регенерации ОЗУ, схемы различных генераторов и т.д.

Гораздо бóльшую разрешающую способность имеют системы диагностирования МПК, сочетающие программно-аппаратную стимуляцию и тестеры, регистрирующие реакцию проверяемой схемы на стимулирующее воздействие, такие, как логические анализаторы, сигнатурные анализаторы и т.д. Это связано с тем, что с помощью этих приборов можно получить доступ к любой точке проверяемой платы или межмашинной магистрали.

Характерной особенностью МПК, существенно усложняющей организацию систем диагностирования, является их переменный состав.

Фирмы–создатели микропроцессорной техники Intel, Zilog, Motorola и др., анализируя потребности пользователей, пришли к выводу о необходимости выпуска не микроЭВМ, имеющих жесткий состав, а наборов различных устройств, предназначенных для выполнения тех или иных функций. Пользователь сам формирует МПК, закупая только те устройства, которые оптимально соответствуют поставленной задаче. В нашей стране такой же подход реализован в микросредствах управляющей вычислительной техники В7(МСУВТ В7) [18], КТС ЛИУС2 [18], СМ 1800 [20]. МПК, создаваемые на базе таких наборов, имеют минимальную стоимость и наиболее полно реализуют требуемые функции.

Устройства, входящие в наборы, в свою очередь, могут не иметь жесткой конфигурации. Состав БИС, их адресация, расположение областей памяти, система прерывания в таких устройствах могут изменяться с целью создания наиболее оптимального МПК. Для изменения конфигурации устройств в них вводятся аппаратные средства адресации. Пользователь, изменяя конфигурацию устройства, приспосабливает его для решения конкретной задачи.

Как показано в [15], проблема организации систем диагностирования вычислительных машин включает в себя: анализ ошибок и их влияния на процесс функционирования машин, формулировку обоснованных технических требований к системам диагностирования, определение функций и выбор состава программных и аппаратных средств для их реализации, точек размещения этих средств в машине, установление связи с основным вычислительным процессом, организацию подготовки

тестовых последовательностей и эксплуатацию оборудования и т.д. Этот подход к проблеме организации систем диагностирования может быть успешно применен к организации систем диагностирования МПК. При этом состав систем диагностирования МПК может быть следующим:

устройство, осуществляющее анализ ошибок;

алгоритмы проверки МПК;

дополнительная аппаратура, вводимая в МПК для организации проверки;

система тестовых последовательностей;

алгоритмы проверки устройств, входящих в состав МПК;

системы контрольных точек;

системы документации, необходимой для осуществления проверки МПК.

Анализ состава системы диагностирования МПК показывает, что изменение конфигурации МПК и входящих в него устройств существенным образом отражается на системе диагностирования. Изменяются алгоритмы проверки как МПК, так и отдельных устройств, система тестовых последовательностей, система контрольных точек, система документации. Выбор устройства, осуществляющего анализ ошибок, также определяется как составом МПК, так и условиями его эксплуатации.

Современные средства диагностирования микропроцессорных устройств. Наиболее мощным оборудованием, предназначенным для поиска неисправностей в микропроцессорных вычислительных устройствах, являются автоматические тестовые системы. Автоматические тестовые системы, например: Columbia 2000 фирмы Columbia Automation [21], MB 7700, MB 7760, MB 7720 фирмы Membrain (Англия), 3050A FAS фирмы John Fluk [22], HP3062A, 3065 фирмы Hewlett Packard [115] (США), АСПД [10] (СССР) и др., представляют собой специализированные мощные вычислительные устройства, позволяющие осуществлять автоматический или полуавтоматический поиск неисправности в печатных платах микропроцессорных устройств.

Автоматические тестовые системы состоят из вычислительной машины с большой емкостью внутренней и внешней памяти, устройства и интерфейса для связи с проверяемой платой (например, через краевые разъемы, через блок внутрисхемной эмуляции, через специальное контактное приспособление в

виде матрицы из подпружиненных игольчатых штырей, обеспечивающих доступ ко всем контактам проверяемой платы и т.д.), устройств доопределения места неисправности (например, сигнатурного анализатора, прибора для определения направления тока и т.д.), мощного специализированного математического обеспечения [24].

Поиск неисправностей с помощью автоматической тестовой системы обычно производится следующим образом: оператор подсоединяет проверяемую плату к автоматической тестовой системе и запускает проверяющую программу. Если при данном отказе можно локализовать неисправность с точностью до компонента схемы, то на экране дисплея отображается информация о месте отказа. Если же при данном отказе невозможна локализация неисправности с требуемой точностью, то на экране дисплея отображается информация о группе подозреваемых компонентов схемы и необходимости доопределения места неисправности, которое производится, например, с помощью сигнатурного анализатора. Последовательность доопределения задается оператору на экране дисплея.

Автоматические тестовые системы позволяют локализовать неисправность в плате за минимальное время с максимальной точностью. При этом обеспечивается обнаружение конкретного отказавшего элемента схемы. Автоматические тестовые системы являются стационарными, дорогостоящими средствами поиска неисправности в сложных микропроцессорных печатных платах и предназначены для их наладки в условиях крупносерийного производства.

Недостатками этих систем являются невозможность поиска неисправности платы в составе микропроцессорного устройства или комплекса, т.е. реальных условиях его работы, а также их высокая стоимость.

Другим мощным средством поиска неисправности являются внутрисхемные эмуляторы, наиболее типичными представителями которых являются внутрисхемные эмуляторы Intellec MDS фирмы Intel [23] и Exerciser фирмы Motorola [25] (США). Внутрисхемные эмуляторы – это микропроцессорные комплексы, в состав которых входят устройства внешней (дисковой) и внутренней памяти, блок внутрисхемной эмуляции, средства отображения (дисплей, принтер), мощное программное обеспечение. Эти устройства предназначены как для поиска неисправностей, так и для отладки программного обеспечения

микропроцессорных устройств. Они же могут использоваться как система автоматического проектирования. При поиске неисправности с использованием внутрисхемного эмулятора из проверяемого устройства вынимается микропроцессор, на его место устанавливается разъем блока внутрисхемной эмуляции, после чего все ресурсы проверяемого устройства оказываются доступными внутрисхемному эмулятору. Так как магистрали данных, адреса и управления проверяемого устройства подключаются к магистралям внутрисхемного эмулятора через буферные шинные формирователи, то это позволяет выполнять диагностическую программу при любом виде отказа в проверяемом устройстве, за исключением отказа генератора тактовых импульсов и блока питания, которые в данном случае становятся ядром оборудования, и их работоспособность должна быть гарантирована.

При поиске неисправности с помощью внутрисхемного эмулятора программно-аппаратным способом определяется неисправное устройство, а дальнейшая локализация неисправной ИС в устройстве производится с помощью осциллографа или логического анализатора. Аппаратная сложность внутрисхемных эмуляторов и сложность математического обеспечения обусловливают высокую стоимость и знание обслуживающим персоналом как аппаратной части, так и программирования.

Внутрисхемные эмуляторы позволяют использовать для хранения тестовых программ собственную внутреннюю или внешнюю дисковую память, разрабатывать программы на языках высокого уровня, исполнять тестовые программы, написанные на этих языках. Развитое матобеспечение предоставляет широкие сервисные возможности.

К недостаткам внутрисхемных эмуляторов относятся:

введение дополнительных тактов ожидания в машинные циклы микропроцессора, из-за чего поиск неисправности производится в режиме, отличном от реального рабочего режима микропроцессорного устройства;

стационарность оборудования, позволяющая производить поиск неисправности в отказавшем устройстве только в составе внутрисхемного эмулятора, а не в составе МПК, в котором оно работает. При этом неисправность не всегда может быть обнаружена – практика зарубежных сервисных фирм показывает, что 50% плат, присылаемых на ремонт, являются исправными, а неисправность проявляется только в составе данного устройства [26];

внутрисхемные эмуляторы предназначены для поиска неисправности только в устройствах, содержащих микропроцессоры.

Логические анализаторы [26–29] – многоканальные приборы, предназначенные для поиска неисправностей в цифровых устройствах, осуществляющие регистрацию, обработку и отображение цифровой информации. Информация отображается в общем случае на экране электронно-лучевой трубки либо в виде квазивременной диаграммы, либо в форме таблицы, представляющей цифровые слова в двоичном, восьмеричном или шестнадцатеричном коде, либо в виде графа. Логические анализаторы позволяют одновременно принимать и обрабатывать информацию с 4, 8 или 16 каналов. Прием информации осуществляется в периоде тактовой частоты проверяемой цифровой схемы. Запуск развертки осуществляется от нужной кодовой комбинации. Могут отображаться данные, полученные как до прихода кодовой комбинации, так и после ее приема.

Логические анализаторы позволяют анализировать квазивременну́ю диаграмму проверяемого устройства и по ее несоответствию правильной временно́й диаграмме определять неисправность. Эти приборы используются не только при поиске неисправности, но и при отладке макетных образцов микропроцессорных устройств (поиске логических ошибок), а также при отладке программ. Особенно полезными логические анализаторы могут быть при работе с устройствами, не имеющими жесткой временно́й диаграммы, например контроллерами дисков и др.

К недостаткам логических анализаторов относятся их высокая стоимость и то, что пользоваться логическими анализаторами могут только достаточно высококвалифицированные специалисты, хорошо разбирающиеся в работе проверяемого устройства и четко представляющие временну́ю диаграмму его работы.

Регистраторы логических состояний [9, 115] – одни из простейших диагностических устройств. Они предназначены для анализа реакции на выходе проверяемого устройства или конкретной ИС на некоторое стимулирующее воздействие, подаваемое на вход тестируемой схемы. Стимулирующее воздействие может генерироваться различными способами: программным; аппаратным или с помощью внешнего

стимулятора [115]. Анализ реакции регистраторами логических состояний производится подсчетом числа:

а) единичных значений в выходной последовательности;

б) переходов из 0 в 1 и из 1 в 0;

в) повторений значений сигналов;

г) фронтов;

д) спадов.

Полученный результат сравнивается с эталонным значением. Однако простой подсчет тех или иных параметров выходного сигнала не может дать высокой достоверности проверки, поскольку легко представить себе отказы, при которых любой из подсчитываемых параметров изменяется, а результат подсчета остается равным эталонному. Поэтому в схему, осуществляющую функцию счета, вводится подсчет с учетом обратной связи с выхода сумматора.

Локализация неисправностей с помощью регистраторов логических состояний наиболее полно исследована и дает максимальный эффект при проверке комбинационных одновыходных схем. Его использование для проверки БИС затруднено, так как любая БИС – это сложная схема с памятью, имеющая множество входов и выходов.

Вероятностное тестирование может производиться по одному из двух алгоритмов:

вероятностное тестирование с использованием эталонного устройства и сравнением результатов воздействия случайных или псевдослучайных входных наборов на эталонное проверяемое устройство;

статистическое тестирование, при котором на вход проверяемого устройства подается множество случайных входных наборов, вычисляется статистическая характеристика реакции проверяемого устройства на входное воздействие и затем производится сравнение полученной статистической характеристики со среднестатистическим значением, которое вычисляется заранее. Результат проверки определяется по отклонению полученной статистической характеристики от среднестатистического значения.

По сравнению с детерминированными методами вероятностное тестирование имеет ряд преимуществ, главным из которых является исключение необходимости вычисления последовательности входных наборов [9]. Однако вероятностное тестирование требует: наличия эталонного устройства, возможности

Таблица 1.4

Тип диагностического устройства	Область применения	Универсальность	Стационарность
Автоматические тестовые системы	Крупносерийное производство	Универсально	Стационарно
Внутрисхемные эмуляторы	Малосерийные заводы-изготовители, лаборатории-разработчики, ремонтные центры	Неуниверсально	Стационарно
Логические анализаторы	Заводы-изготовители, лаборатории-разработчики, ремонтные центры, эксплуатация	Универсально	Полустационарно
Приборы, осуществляющие подсчет параметра проверяемой схемы	Эксплуатация	Неуниверсально	Нестационарно
Сигнатурные анализаторы	Заводы-изготовители, ремонтные центры, эксплуатация	Универсально	"
Устройство поэлементного контроля в виде клипс	Доопределение неисправности	Неуниверсально	"

Примечание. *За единицу берется стоимость сигнатурного анализа.

Стоимость (в условных единицах *)	Разрешающая способность	Возможность поиска неисправности в устройстве на месте эксплуатации	Квалификация обслуживающего персонала	Поиск неисправности в реальном масштабе времени	Наличие собственного программного обеспечения
200	До неисправного компонента схемы	Нет	Оператор	Да	Есть
20	До неисправного устройства	”	Инженер	Нет	”
10	До неисправного компонента схемы	Затруднен	”	Да	Нет
1	До компонента схемы	Есть	Оператор	”	”
1	То же	”	”	”	”
1	” ”	”	”	Нет	”

его присоединения к проверяемому устройству, аппаратуры для генерации входных случайных наборов и анализа реакции на них. Кроме того, поскольку БИС, входящие в состав МПК, являются сложными программируемыми устройствами, то подача случайных входных приборов на их входы может привести к подаче на них запрещенных входных комбинаций, реакция на которые непредсказуема.

Сигнатурные анализаторы (СА) [26, 30–36] представляют собой цифровые устройства, позволяющие осуществлять с высокой точностью сжатие входной двоичной последовательности в короткий ключевой код – сигнатуру. Поиск неисправности с использованием сигнатурных анализаторов заключается в сравнении сигнатуры, полученной с вывода проверяемой ИС, и эталонной сигнатуры, записанной в словаре эталонных сигнатур, т.е. в сравнении двух чисел. Сигнатурные анализаторы позволяют осуществлять поиск неисправности в реальном масштабе времени, поскольку генерация стимулирующих воздействий на проверяемое устройство обычно осуществляется самим проверяемым устройством. Небольшие размеры и масса сигнатурных анализаторов позволяют производить поиск неисправности непосредственно на месте установки проверяемого оборудования. Простота поиска неисправности позволяет оператору, не имеющему высокой квалификации, успешно локализовать неисправность.

К недостаткам сигнатурного анализа относятся: невозможность поиска неисправности в схемах с обратной связью, необходимость адаптации на этапе разработки микропроцессорных устройств к поиску неисправности методом сигнатурного анализа, большое количество сопроводительной документации, обеспечивающей поиск неисправности.

Анализ автономных средств поиска неисправностей произведен по табл. 1.4, из которой видно, что сигнатурные анализаторы являются одним их самых простых, дешевых и универсальных средств поиска неисправностей в микропроцессорных устройствах и могут быть использованы как в производстве, так и эксплуатации микропроцессорных вычислительных устройств.

1.2. НЕИСПРАВНОСТИ В ЦИФРОВЫХ УСТРОЙСТВАХ

Прежде чем начать что-то искать, нужно понять, что вы собираетесь искать. В литературе по диагностике чаще всего рассматриваются неисправности в ИС и ЭРЭ цифровых устройств. Однако существует большая группа неисправностей, связанных с элементами устройства, обеспечивающими электрическое соединение между выводами ЭРЭ и ИС.

Конструкции цифровых устройств состоят из:

1) многослойной или двусторонней печатной платы, содержащей печатные проводники и металлизированные отверстия, на которой монтируются ЭРЭ и ИС или соединители под них, а также соединители, обеспечивающие установку платы в шасси устройства и связь с внешними устройствами;

2) многослойной или двусторонней печатной платы объединительной панели, которая позволяет обеспечить электрическую связь между платами устройства и подвод к ним питающих напряжений; на плате объединительной панели монтируются ответные части соединителей, установленных на печатных платах;

3) системы кабелей, которые служат для связи с внешними устройствами и объектом управления.

Кроме перечисленных частей в конструкцию микроУВК входит еще множество различных элементов: кожухи, вентиляторы и т.д., но, поскольку они не имеют отношения к функциональным неисправностям, эти элементы не будут рассматриваться.

Неисправность может быть охарактеризована с двух точек зрения: с электрической – рассматриваются электрические параметры неисправности, и конструктивной – определяется физическое место неисправности и микроЭВМ.

Электрические параметры неисправности. Любая вычислительная машина представляет собой сложную электрическую схему, в состав которой входят линейные и нелинейные электрические приборы, а именно: резисторы, конденсаторы, диоды, транзисторы, кварцевые генераторы, интегральные схемы (которые, в свою очередь, являются совокупностью перечисленных выше более простых электрических приборов), светодиоды и т.д.

В общем случае в такой электрической цепи, если не рассматривать ситуации, связанные с некорректным проектированием, могут возникать неисправности следующих видов:

1. Короткое замыкание (КЗ). В эту группу неисправностей входит множество различных видов по характеру КЗ: короткое замыкание сигнального проводника с шиной питания, земляной шиной, с другим сигнальным проводником и т.д.

2. Обрыв проводника. В эту группу входят все неисправности, приводящие к разрыву электрической цепи как сигнальных шин, так и шин питания.

3. Выход электрических приборов схемы на нерасчетные режимы работы, например в результате старения или потери герметичности, т.е. под действием факторов, не связанных с первыми двумя видами неисправности.

4. Неисправности, связанные с временны́ми параметрами, когда вследствие неправильного проектирования или нарушения временны́х характеристик элементов происходит запаздывание одного сигнала относительно другого и в результате – прием ошибочной информации.

Рассмотрим поведение цифрового устройства при различных видах неисправностей для наиболее распространенного в настоящее время вида интегральных схем, использующих транзисторно-транзисторную логику (ТТЛ). В ИС этого типа используются, как правило, два уровня напряжения питания +5 В и "Земля". Аналогичное питание используется и в БИС К580ИК55, которая выполнена с использованием МОП-технологии.

Короткое замыкание. Короткое замыкание (КЗ) между шинами питания приводит к автоматическому отключению блока питания и легко определяется измерением сопротивления между различными шинами питания.

КЗ между шиной с земляным потенциалом и сигнальной линией приводит к установлению на этой сигнальной линии потенциала, близкого к 0 В. При исправной ИС, выход которой имеет КЗ с земляной шиной, сигнал на ее выходе будет иметь некоторое отклонение от 0 В в виде выбросов на фронте и спаде выходного сигнала, формируемого в соответствии с реализуемой функцией (рис. 1.9–1.12). В общем случае КЗ сигнального проводника с земляной шиной всегда приводит к подавлению сигналов, имеющих уровень Лог. 1, т.е. имеющих потенциал 2–4,8 В.

КЗ между шиной питания с потенциалом +5 В и сигнальной линией приводит к установлению на этой сигнальной линии потенциала, равного потенциалу шины питания. Сигнал на

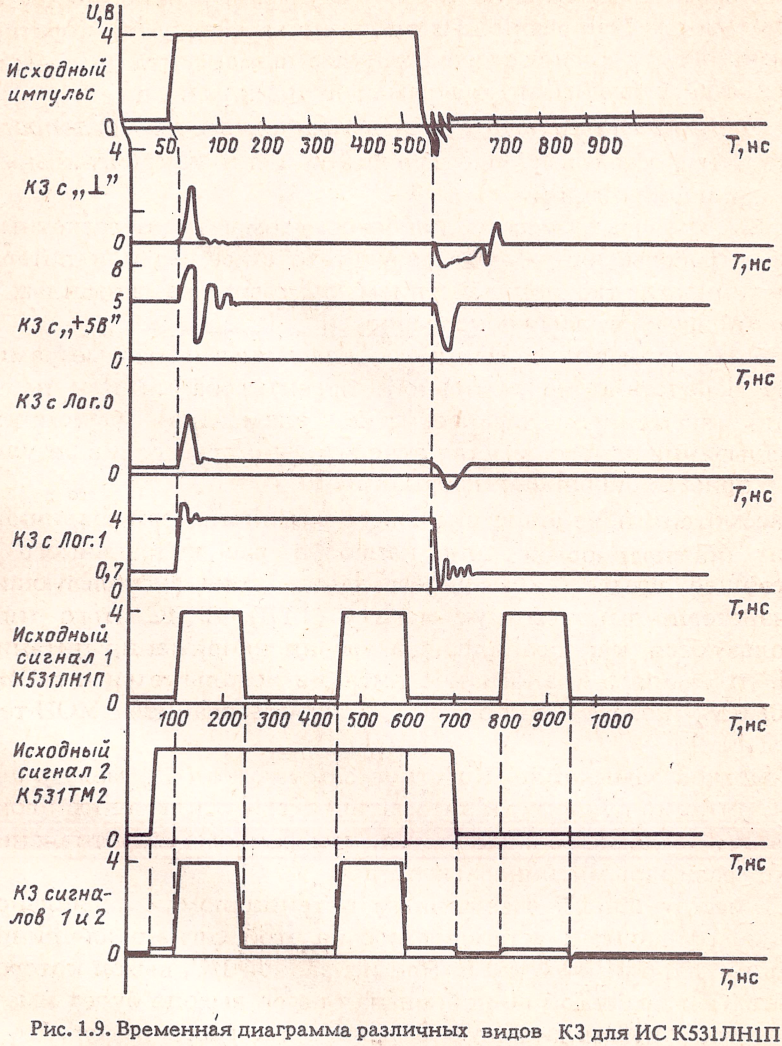


Рис. 1.9. Временная диаграмма различных видов КЗ для ИС К531ЛН1П

этой линии будет иметь выбросы на фронте и спаде сигнала, формируемого в соответствии с реализуемой функцией (рис. 1.9–1.12).

КЗ между двумя сигнальными линиями приводит к реализации функции монтажного И между этими сигналами (рис. 1.9–1.12). Потенциал Лог.0 всегда доминирует над потенциалом Лог. 1.

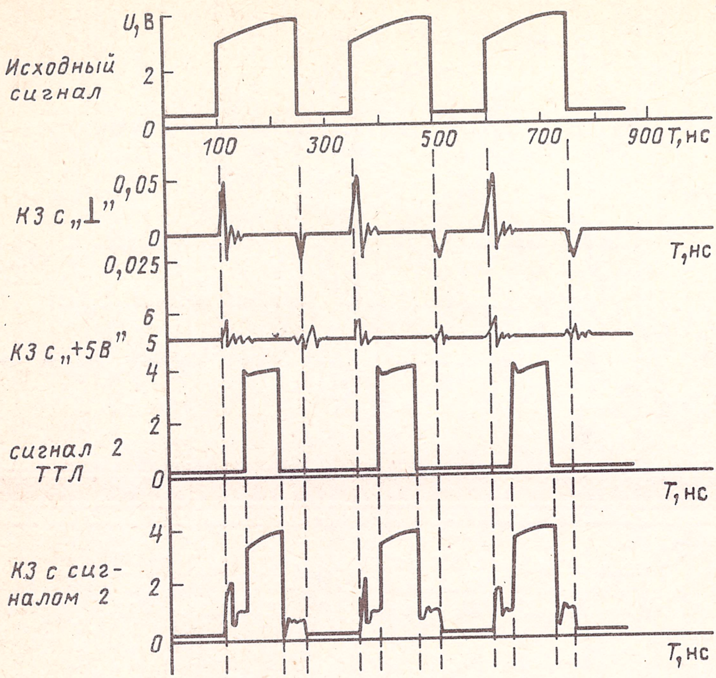


Рис. 1.10. Временные диаграммы различных видов КЗ для ИС К155ЛП8

К группе отказов типа КЗ относятся также отказы, связанные с пробоем электрических приборов, при которых сопротивление между его выводами становится близким нулю.

КЗ являются наиболее сложными для поиска видами неисправностей. Некоторые методы их локализации будут приведены ниже.

Обрыв проводника – один из наиболее распространенных видов неисправностей. При обрыве проводника, подводящего напряжение питания на ИС, на выходах схем устанавливаются потенциалы, показанные на рис. 1.13.

Обрыв проводника, связывающего ИС с земляной шиной, приводит к ситуации, показанной на рис. 1.13.

При обрыве сигнального проводника, если только этот обрыв произошел не внутри ИС, вывод ИС, являющийся выходом, будет работать нормальным образом, а на выводах ИС, являющихся входами этого сигнала, установится потенциал, приблизительно равный 1,4 В.

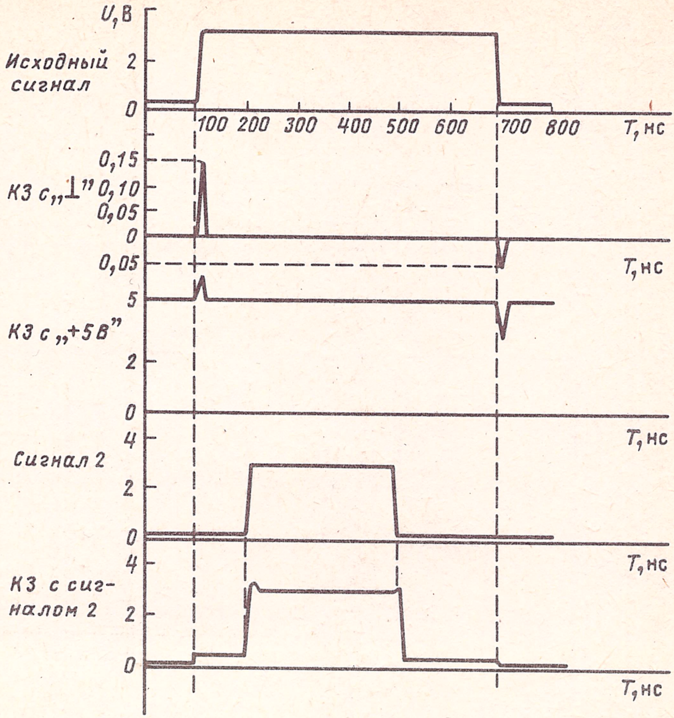


Рис. 1.11. Временная диаграмма различных видов КЗ для ИС К589АП16

Как было показано выше, любая вычислительная машина, в том числе и микроЭВМ (УВМ), является сложной конструкцией, каждый элемент которой обладает своей собственной надежностью и может явиться причиной неисправности.

Как известно любому человеку, занимающемуся наладкой или эксплуатацией такой техники, наиболее частой причиной неисправности являются ИС. Это связано не с малой степенью их надежности, а с тем, что количество ИС различного типа и различной степени интеграции не идет в сравнение с количеством любых других элементов, входящих в состав цифровых устройств.

Неисправности в ИС могут быть любого из трех типов: КЗ, обрыв проводника, выход ИС из режима. Причины неисправностей могут быть различны: некачественная сборка ИС, в том числе и некачественная пайка выводов; нарушение технологии при изготовлении кристалла; разгерметизация корпуса и после-

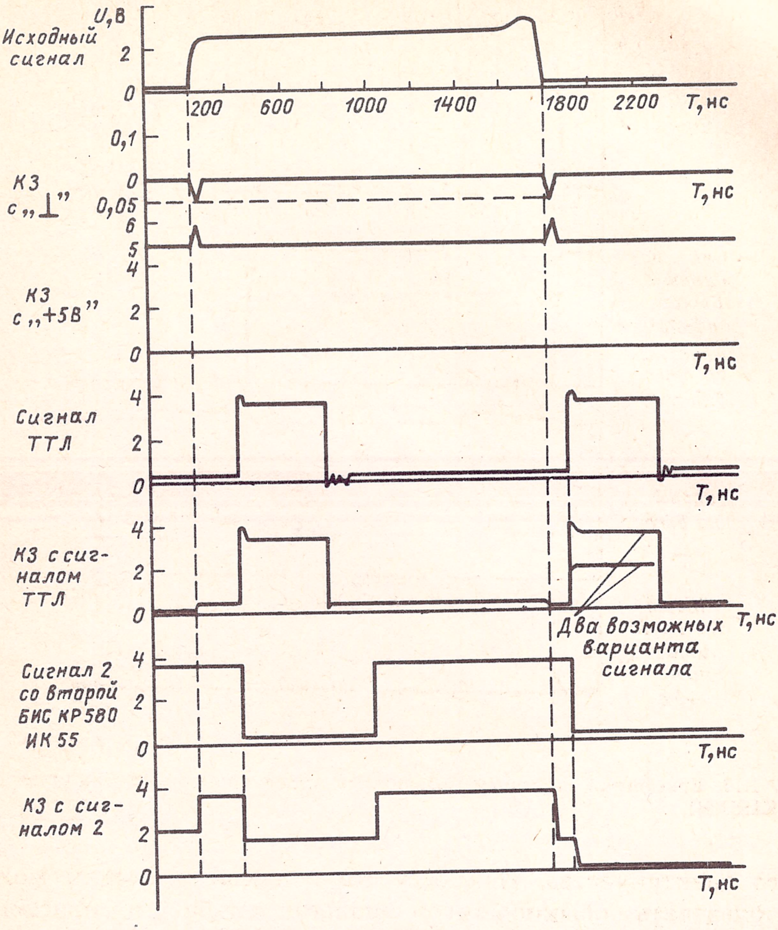


Рис. 1.12. Временная диаграмма различных видов КЗ для БИС К580ИК55

дующее проникновение влаги и грязи внутрь ИС, что приводит к возникновению процесса коррозии; электрический пробой активных элементов ИС под воздействием электростатического заряда. Особенно чувствительными к ним являются ИС, выполненные по МОП-технологии. Для предотвращения неисправностей последнего типа требуется аккуратность в обращении с ИС, не только установленными в устройство, но и находящимися на хранении. Хранение, транспортировку и переноску ИС необходимо осуществлять в специальной упаковке, препятствующей накоплению на ней статичес-

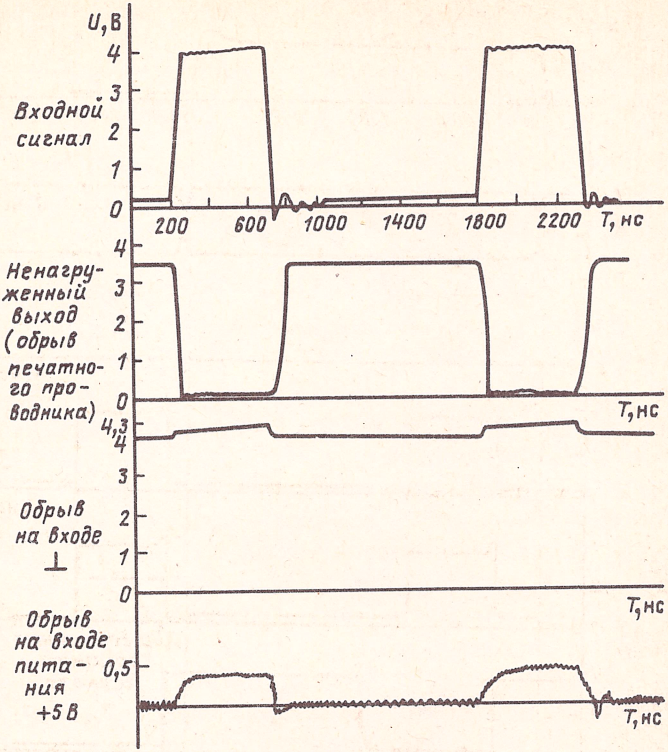


Рис. 1.13. Временная диаграмма различных видов обрывов проводников для ИС К155ЛН1

кого электричества. При отсутствии такой упаковки можно использовать обыкновенную фольгу, сделав в пластинке фольги два желоба, в которые вставляются выводы ИС. Нужно обратить внимание на то, что современные БИС выполнены, как правило, на МОП-технологии и только по выходным сигналам и напряжениям питания совместимы с биполярным ТТЛ ИС, например: ПЗУ серии К573, микропроцессорный комплект К580, К1810 и др.

ИС устанавливается на многослойную печатную плату двумя способами: непосредственно установкой выводов ИС в металлизированные отверстия, сделанные в плате, и последующей их распайкой или установкой на соединитель, который запаян в такие отверстия. Эти соединения также могут служить источником неисправностей, причем именно эти соединения часто

являются источником самых неприятных неисправностей – сбоев.

Неисправности, возникающие в соединении вывода ИС и платы или соединителя, относятся к контактным неисправностям – обрывам.

При установке ИС в соединитель электрическая связь между выводами ИС и контактом соединителя осуществляется механическим способом за счет упругих свойств контакта. Существуют различные типы соединителей, в которых обеспечивается односторонний или двусторонний контакт. При одностороннем контактировании пружина прижимает вывод ИС к запаянной в плату пластине соединителя. В этом случае электрический контакт осуществляется в одной плоскости, поскольку пружина является изолированным элементом конструкции соединителя. В соединителях с двусторонним контактированием оба элемента конструкции контакта соединителя являются проводниками и соединены с печатной платой, за счет этого надежность таких соединителей выше. Однако как первый, так и второй виды соединителей рассчитаны на определенные количества сочленений, и частая смена ИС ведет к разрушению контактов соединителя и потере электрической связи. Металлические элементы конструкции со временем теряют свои упругие свойства, что также приводит к возникновению неисправности. Для таких соединителей необходимы точность изготовления отдельных элементов конструкции и обеспечение качества сборки.

Как было показано выше, неисправность может возникнуть в любом месте электрической цепи, при этом неисправности в ИС могут возникать как в выходных, так и во входных цепях. Поэтому необходимо понимать, что большинство существующих в настоящее время методов диагностики не позволяет локализовать неисправность с точностью до неисправного компонента схемы. В лучшем случае будет определена неисправная цепь схемы, состоящая из одного или нескольких выходных и входных контактов ИС и других ЭРЭ.

После определения неисправной цепи схемы необходимо локализовать неисправность, т.е. выявить точное место неисправности. Наиболее распространенным видом отказа является отказ в выходном каскаде ИС, являющимся генератором сигнала в данной цепи. Поэтому, как правило, при поиске неисправности сразу осуществляют проверку работоспособности таких ИС. Эта

проверка может быть осуществлена различными способами: специальным тестером, определяющим работоспособность отдельной ИС без ее выпаивания из платы; этим же тестером могут быть проверены и другие ИС цепи, служащие как приемник сигнала; механическим разрывом цепи, приводящим к изоляции отдельных ее участков и анализу работоспособности изолированного участка с помощью осциллографа или другого измерительного прибора, однако такой вид контроля приводит к разрушению выводов ИС или печатных проводников.

Если проведенный анализ неисправности показывает, что ее источником являются элементы печатной платы, то методы поиска неисправности и способы ее устранения зависят от характера неисправности.

Наиболее простым случаем является разрыв электрической цепи. Здесь даже может не требоваться точное определение места разрыва, что необходимо только при проведении специальных исследований с целью определения слабых мест или выявления источника брака, а достаточно простого восстановления разорванной цепи с помощью проводных перемычек, заменяющих поврежденный участок цепи.

При возникновении КЗ поиск неисправности существенно затруднен, особенно в том случае, когда неизвестна вторая цепь, с которой возникло КЗ.

Если же известны обе цепи, между которыми существует КЗ, то поиск места КЗ можно производить с помощью специального прибора, например типа Щ. 4315-03. Этот прибор позволяет производить измерение электромагнитного поля, существующего вокруг проводника. На рис. 1.14 приведена схема, поясняющая этот метод определения места КЗ. Поиск неисправности производится следующим образом: выходы генератора импульсов соединяются с обеими цепями (рис. 1.14, *а*). После этого откалиброванный прибор переводится в режим измерения и оператор прослеживает печатный проводник одной из соединенных цепей до того момента, пока прибор не перестанет показывать наличие тока в проводнике. Место короткого замыкания уточняется путем повторных измерений после перестановки места подсоединения генератора (рис. 1.14, *б,в*).

Если цепь, в которой произошло КЗ, неизвестна, то необходимо ее найти. Для определения двух замкнутых сигнальных цепей нужно предварительно исследовать топологию печатной платы по ее документации (фотоотпечаткам). При этом

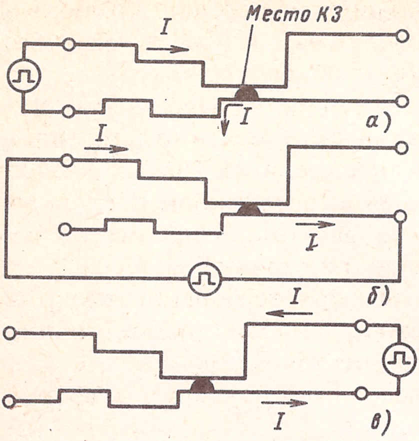


Рис. 1.14. Иллюстрация поиска места КЗ между печатными проводниками

просматривается прохождение неисправного печатного проводника и определяются все рядом проходящие печатные проводники, контактные площадки межслойных переходов и площадки, в которые запаиваются ИС. После этого производится включение устройства и запуск теста, на котором была выявлена неисправность. Дальнейший поиск цепи, в которой произошло замыкание, можно производить любым диагностическим прибором (осциллографом, логическим или сигнатурным анализатором), последовательно проверяя сигналы во всех цепях, определенных во время анализа топологии платы. Определение замкнутой цепи с помощью осциллографа или логического анализатора производится путем проверки на идентичность сигналов в двух цепях: неисправной и подозреваемой, а сигнатурный анализатор покажет в проверяемой цепи ту же сигнатуру, что и в неисправной. Подтверждение замыкания осуществляется с помощью омметра. Поиск места замыкания в двух цепях проводится по методике, описанной выше.

1.3. ОБЩИЕ ВОПРОСЫ ОТЛАДКИ И ДИАГНОСТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Задачи отладки и диагностирования возникают на всех стадиях жизненного цикла микропроцессорных систем (МПС) – разработки, изготовления и эксплуатации.

Процесс отладки состоит в определении правильности ряда соотношений в аппаратуре и программах – параметрических (задержки, уровни сигналов и пр.), логических, функциональных и конфигурационных соотношений с последующим устранением имеющихся нарушений [1, 2].

В процессе отладки можно выделить следующие компоненты – объект отладки (ОТ), средства отладки (СО) и внешнюю среду (ВС), с которой взаимодействует объект отладки.

В качестве объекта отладки выступают аппаратные и программные компоненты МПС. В качестве средств отладки применяются аппаратные и программные средства. Процесс отладки состоит в общем случае из подачи определенных входных воздействий, получения выходных реакций, принятия решения по этим реакциям, устранения нарушений.

Объект отладки характеризуется приспособленностью для целей отладки, может определяться числом доступных для проверки входов и выходов, внутренних точек [3].

Таким образом, процесс отладки можно представить состоящим из таких крупных этапов:

подготовка тестовых (отладочных) воздействий и их программирование;

автономная отладка оборудования и программ;

комплексная отладка оборудования и программ.

Особенно важное значение имеет расчет эталонных значений сигналов для отдельных узлов и блоков, в частности для БИС микропроцессоров, что требует разработки специальных программ моделирования.

Процесс отладки содержит ряд типичных диагностических операций, в связи с чем известные методы тестирования могут быть применены в процессе отладки. Одновременно возникают задачи автоматизации процессов отладки и диагностирования МПС и выбора наиболее рациональных вариантов их организации.

Вначале рассмотрим особенности отладки компонентов в МПС.

Задачи отладки и диагностирования, как отмечалось, возникают на этапе проектирования, при разработке первого образца новых устройств, машин и систем.

Эти же задачи возникают в процессе производства, при установке и эксплуатации систем и машин с применением МПС.

В отличие от обычных задач диагностирования при отладке на всех стадиях жизненного цикла МПС приходится иметь дело с широким спектром дефектов – ошибками проектирования, изготовления, дефектами вследствие отказов.

На этапе проектирования и изготовления возникают ошибки при исходном задании алгоритма, при синтезе, при изготов-

лении чертежей, описании связей для разводки, при ручной разводке, вследствие дефектов фотошаблонов, вследствие ошибочных действий монтажника, пропуска бракованных схем на входном контроле. В процессе эксплуатации возникают ошибки вследствие отказов различных элементов, вызванных деградацией физико-химических свойств материалов, из которых они изготовлены. Большой класс ошибок вызывается дефектами, изменяющими динамические параметры.

Наиболее массовыми являются ошибки синтеза (до 40%), что объясняется большим количеством ошибок в синхронизации регистровых схем и организации связей между различными функциональными узлами.

Исходя из задач отладки, можно сформулировать общие требования к наладочному оборудованию, которые состоят в следующем:

способность заменять отсутствующие блоки и устройства за счет интерпретации их режимов работы;

способность быстро устанавливать программно-физическую связь с каждым устройством;

удобное и оперативное формирование тестовых последовательностей;

имитация необходимых режимов работы оборудования в статике и динамике;

возможность оперативного анализа результатов тестирования и визуализации их в комплектной и удобной для пользователя форме;

возможность оперативной модификации тестов в процессе отладки;

оперативное вмешательство в процесс отладки для выяснения причины неправильного функционирования и ее устранения;

одновременное наблюдение за большим числом параметров.

Разработка микропроцессорных систем и отладочные комплексы. В общем случае микропроцессорные системы включают в себя технические средства, основой которых является микропроцессорная аппаратура (МПА) и программное обеспечение (ПО), внешнее по отношению к МПА.

МПА обычно состоит из схемного оборудования и внутренних (встроенных) программ или микропрограмм.

Таким образом, в МПС отладке подлежат программно-микропрограммная и схемная части, или то и другое вместе (комплек-

сная отладка). Затраты на разработку программно-микропрограммной части являются значительными и могут составлять от 60 до 90% общих затрат [1].

В настоящее время наряду с таким классом МПС, как микровычислительные комплексы (МВК) типа персональных ЭВМ (ПЭВМ), проблемно-ориентированных вычислительных комплексов, получили распространение УВК и микропроцессорные устройства (МУ), предназначенные для использования в реальном времени для управления различными системами и изделиями. Этот класс систем является чрезвычайно важным, так как к нему относится практически вся управляющая техника.

Зачастую МУ являются встроенными в объект управления и не имеют традиционного периферийного оборудования и внешних запоминающих устройств, программы и микропрограммы в таких устройствах, как правило, хранятся в постоянных запоминающих устройствах (ПЗУ). Обычно к таким МУ предъявляются высокие требования по надежности, что требует разработки специальных мер – применения резервирования и т.д.

Обычно для ускорения разработки МУ применяют специальные отладочные комплексы, включающие средства отладки в реальном времени. Такие комплексы повышают производительность труда разработчиков в 2–3 раза, и зачастую их называют автоматизированными рабочими местами (АРМ).

Выделяют два типа АРМ – с внутренним и внешним макетированием. В первом случае отлаживаемое устройство одновременно является физическим макетом и инструментальным прототипом (прототипные комплексы).

АРМ с внешним макетированием включает в себя специальную аппаратуру. К базовой ЭВМ присоединяются следующие инструментальные средства: внутрисхемный эмулятор (ВСЭ), программатор, логический анализатор и имитатор внешней среды.

Примерами таких комплексов являются АРМ для программистов, АРМ для микропрограммистов, комбинированные АРМ.

В настоящее время промышленностью выпускается ряд отладочных комплексов. К ним относятся: система СО-04 для отладки МПС, разрабатываемых на основе микропроцессора К580ИК80 (относится к типу АРМ для программиста).

Примером комбинированного АРМ (для программирования и микропрограммирования) является АРМ-2-05.1 [5].

Перечисленные комплексы относятся преимущественно к этапу проектирования, однако они могут при определенных изменениях использоваться также при производстве. Применение их на площадке заказчика является проблематичным.

Как для использования в составе отладочных комплексов (на этапе разработки), так для автономного использования (для сопровождения и работы на площадке заказчика) необходимо выполнять ряд диагностических операций, обеспечивать обработку диагностической информации в реальном времени.

Из существующих методов наиболее эффективными являются методы логического и сигнатурного анализа и построенные на их основе инструментальные средства – логические и сигнатурные анализаторы.

Их основное достоинство – возможность в реальном времени обрабатывать диагностическую информацию о динамических, структурных и функциональных свойствах как схемных, так и программных компонентов МПС.

Однако их применение требует выполнения определенных подготовительных операций на отлаживаемом устройстве – выбора средств стимуляции, точек подключения.

Сложность дополнительных операций определяется подготовленностью самого объекта отладки и во многом определяет трудоемкость процесса отладки.

Весь этот комплекс вопросов входит в понятие организации технологического процесса отладки.

Средства отладки при эксплуатационном обслуживании МПС. В настоящее время средства МПС, отличающиеся большой сложностью, стали неотъемлемой частью установок для обработки данных, связи, комплексной автоматизации практически во всех отраслях. При этом возникает объем работ, связанных с монтажом оборудования, отладкой, его техническим обслуживанием и ремонтом в процессе эксплуатации. (Например, объем рынка эксплуатационного обслуживания в США возрастет в 1990 г. в 2 раза по сравнению с 1984 г. – с 20 до 46 млрд. дол. Стоимость схемных плат, проделавших путь от заказчика на ремонт и обратно, превышала в 1984 г. 10 млрд. дол.).

Имеются следующие организационно-технологические формы обслуживания, связанные с ремонтом: обслуживание на месте эксплуатации оборудования, обслуживание в центре ремонта, самообслуживание силами заказчиков и обслуживание в центрах, производящих не ремонт, а замену.

Основные формы общего обслуживания – индивидуальное, централизованное (в том числе дистанционное и смешанное). Прогрессивным является комплексное обслуживание, дистанционная диагностика.

Дистанционная диагностика снижает стоимость ремонта и значительно сокращает число специалистов высокой квалификации, требуемых для обслуживания новой техники.

Организация дистанционной диагностики подразумевает простую оперативную связь (например, по телефону) между техником на месте и специалистами из обслуживающего центра. Для идентификации неисправностей создаются базы данных, которые используются для передачи указаний по телефону для ускорения обслуживания. Следующим шагом является использование систем искусственного интеллекта для моделирования процесса отыскания неисправностей. Дистанционная диагностика может обеспечивать режим не только совета, но и непосредственного диагноза на месте заказчика, для чего используются логические анализаторы и интерфейс IEEE-488, RS-232C.

В центрах обслуживания устанавливается автоматизированное испытательное оборудование, способствующее повышению производительности, снижению запасов технических элементов замены (ТЭЗ) для ремонта.

Разработка концепции эксплуатационных и аппаратно-программных средств поддержки эксплуатации является важной составной частью общего процесса проектирования.

В настоящее время общепринятой является концепция, согласно которой современные средства ВТ должны снабжаться комплексом аппаратурно-программных средств поддержки эксплуатации, удобно сочетающихся со структурой этих средств.

Выделяются такие системы поддержки:

система оперативного автоматического контроля, которая выполняет функции обнаружения ошибок и предотвращения их распространения по программе и искажения данных путем прерывания вычислительного процесса; эта система освобождает пользователя от необходимости контроля правильности работы;

система оперативного восстановления вычислительного процесса, на которую возлагается автоматическое повторение неправильно выполненных операций, участков программы и таким путем обеспечивается защита от сбоев;

система автоматического поиска неисправностей, включаю-

щая комплекс средств и методик для обеспечения и ускорения этапа поиска неисправностей, сокращения времени и труда на выполнение этих операций;

система автоматической регистрации и обработки данных об отказах и сбоя с целью адаптации МПС к условиям эксплуатации, обоснованного планирования профилактик, ремонтных работ, доработок;

система автоматической программно-управляемой профилактики, обеспечивающая автоматические изменения уровней вторичных питающих напряжений, частоты и формы управляющих сигналов, выдачу сообщений об элементах, параметры которых находятся вблизи критических значений.

Эти системы взаимодействуют между собой, дополняя друг друга в процессе эксплуатации.

1.4. ОРГАНИЗАЦИЯ ПРОЦЕССА ОТЛАДКИ И ОТЛАДОЧНЫЕ ОПЕРАЦИИ

Под организацией процесса отладки понимают выбор комплекса технических средств получения и обработки диагностической информации, точек подключения к объекту, способов синхронизации и методик использования применительно к условиям различных стадий – разработки, производства и эксплуатации.

Особенное значение имеет организация комплексной отладки оборудования и программ, где проверяются не только логические, но и динамические соотношения. Процесс отладки в этом случае имеет много общего с процессом тестирования МПС, и поэтому можно использовать некоторые общие процедуры, разработанные для этой цели, в частности, типа процедуры расширяющихся областей, при которой в составе МПС выделяется предварительно проверяемое ядро. Затем проверенное ядро используется для стимуляции следующего устройства и т.д.

В рамках таких общих схем организации тестирования и отладки могут применяться следующие процедуры диагностирования:

процедура "дерево", при которой поиск начинается с большого числа функциональных блоков и на каждом шаге проверяется успех либо неуспех прохождения теста. В случае неуспеха производится переход к более подробном анализу

подозреваемых блоков. Эта процедура соответствует реализации условных алгоритмов диагностирования;

процедура "начиная с малого", при которой проверяется вначале малая часть, а затем с небольшими приращениями остальная часть машины;

процедура "пересечения", отвечающая безусловным алгоритмам поиска, при которой после анализа серии результатов тестов производится выделение подозреваемых в неисправности блоков.

Рассмотрим типичные операции общего характера, связанные с реализацией непосредственно процесса отладки.

К ним относятся:

подготовка к работе отладочных средств и объекта;

знакомство с документацией по объекту отладки, получение тестов;

подключение отладочных средств к объекту;

задание режимов работы объекта и отладочных режимов;

автоматизированный сбор данных о состоянии объекта;

анализ и обработка полученных данных;

отображение их в удобной форме (вторичный анализ);

выдача корректирующих и управляющих воздействий.

Каждая из этих операций, в свою очередь, состоит из ряда более мелких и требует определенных затрат времени, квалификации персонала и характеризуется определенным уровнем трудоемкости и эффективности.

Особо следует обратить внимание на операции, выполняемые вручную или полуавтоматически, так как кроме снижения производительности по сравнению с автоматизированным режимом при их выполнении вносятся еще ошибки человека-оператора. В первую очередь это относится к заданию режимов и параметров, определяющих функционирование объекта отладки. Целесообразно для этого использовать специальные укрупненные формализованные описания режимов и параметров, которые помимо повышения производительности снижают число ошибок.

В качестве таковых применяют языки высокого уровня типа Бейсик, специальные языки либо используют работу в режиме "меню", что тоже очень удобно для обслуживающего персонала невысокой квалификации. Таким образом, программирование отладочных операций, в частности, для логического анализатора, является важным средством повышения производительнсоти труда при его использовании.

Как отмечалось, отладка состоит в подаче определенных стимулирующих воздействий (наборов кодов, команд, их последовательности) и наблюдении в заданное время выбранных (в соответствии с общими стратегиями отладки) определенных диагностических функционалов. Формирование таких функционалов осуществляется с помощью отладочных операций. Это следующие операции:

определение времени выполнения отдельных команд и сравнение его с ожидаемым;

связывание логических ошибок, действие которых проявляется через большой отрезок времени;

определение частоты использования команд "стоп-кадр" и повтор одного или нескольких участков по времени, коду команды и другим признакам;

проверка адресации;

идентификация и анализ полей микрокоманд;

просмотр и идентификация отдельных данных и массивов данных;

отлеживание программы от начала до любой точки;

перечисление маршрутов программ;

определение состояния устройств (активно, пассивно, какую операцию выполняет);

определение времени ответа устройств.

Операции анализа и отображения диагностической информации предназначены для выполнения следующих функций:

формирования различного рода представлений (карт, таблиц, гистограмм, ломаных кривых, диаграмм);

представления длительностей выполнения программ команд, микрокоманд;

распределения информации в таблицах по разным признакам (видам команд, по взаимному следованию, процентному соотношению);

использования масштабирования и комбинации масштабов;

частоты использования модулей, команд, микрокоманд;

построения профилей работы отдельных устройств.

Выделяют также класс активных отладочных действий, связанных с вмешательством оператора в процесс автоматической отладки. К ним относятся:

установ регистров в определенное состояние,

замена адресов,

замена содержания регистров,

исключение операторов в программе,
добавление операторов в программу,
повтор одного или нескольких участков по времени,
изменение синхронизации.

Перечисленные функциональные операции требуют для своей реализации специальных и программных средств. Наиболее приспособленными для реализации диагностических операций, связанных с определением функционалов, являются логические и сигнатурные анализаторы. Наиболее полным объемом отладочных операций обладают системы, состоящие из сочетания логического анализатора с персональным компьютером [6]. Такие системы в настоящее время получают самое широкое применение.

Подробные рекомендации по выбору программных или аппаратных средств дать сложно, однако можно высказать следующие общие соображения. Наиболее эффективна обработка диагностической информации в реальном времени. Поэтому выяснение того, что произошло, безусловно предпочтительнее выполнять быстродействующими аппаратными средствами с малым запаздыванием и без накладных расходов времени. Выяснение, почему произошло, можно выполнять программно. Причинно-следственные отношения, таким образом, можно выяснять на программном уровне. Различают следующие общие приемы отладки: трассировка команд, использование принципа перехвата, принцип выборочных замеров и др.

1.5. ТЕСТОВОЕ ДИАГНОСТИРОВАНИЕ УЗЛОВ, БЛОКОВ И УСТРОЙСТВ МПС

Проверка отдельных узлов и блоков осуществляется на стадии производства и эксплуатации МПС с помощью специального оборудования, не входящего в состав основного оборудования машины. В качестве объекта диагностирования (ОД) выступают функциональные устройства – арифметические устройства (АУ), процессор, ОЗУ, УВВ, законченные ТЭЗы и корпуса СИС, БИС, СБИС. Автоматизация проверок позволяет повысить достоверность и производительность операций контроля [7].

При проверках отдельных схем и элементов выделяют три типа тестирования:

статическое – частота смены тестовых наборов на входе проверяемого устройства и частота съема реакций значительно ниже, чем при работе устройства в реальных условиях;

динамическое – входные наборы подаются, а выходные реакции анализируются на частотах, максимальных для данного устройства;

параметрическое – проверяются динамические параметры и предполагаются измерения уровней напряжения и тока, задержек и других параметров.

Для тестового диагностирования узла и блока необходимо выполнить следующие операции: подготовить тестовые воздействия и эталонные реакции, перенести их на носители информации; сформировать и подать тестовые воздействия на ОД (этап стимуляции), снять и зарегистрировать выходные реакции и провести их оценку путем сравнения с эталонными (этап анализа).

Выходные воздействия и эталонные реакции могут формироваться до проверки и в ходе ее (т.е. в реальном времени).

Подача воздействий и съем реакций может осуществляться в различных точках – через краевые разъемы плат, через разъем микропроцессора, во всех точках проверяемого устройства.

Каждая из диагностических операций может выполняться на различных принципах с помощью различных средств, что и определяет основные функциональные схемы организации тестового диагностирования, показанные на рис. 1.15.

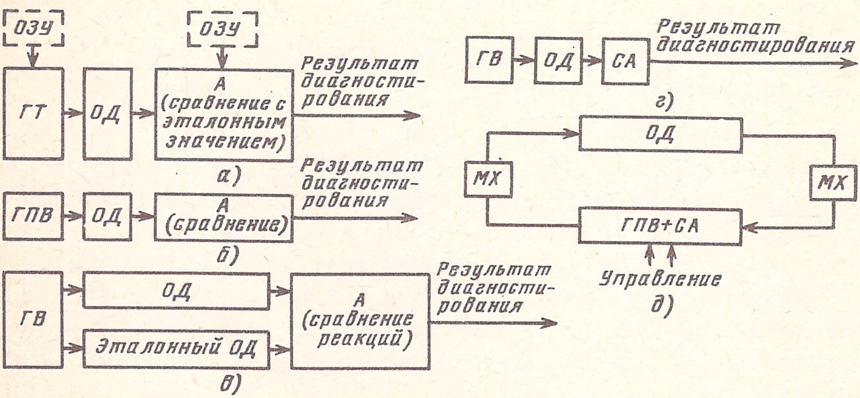


Рис. 1.15. Основные функциональные схемы организации тестового диагностирования

По методам стимуляции и получения оценки реакции проверяемого устройства выделяют следующие типы тестирования.

Тестирование с хранимой программой. Функциональная схема организации показана на рис. 1.15, *а* и включает генератор тестовых воздействий (ГТ), содержащий набор статистических тестов, подготовленных заранее вручную или автоматически, ОД и анализатор (А), работающий по принципу сравнения выходной реакции с эталонной, полученной также заранее специальными средствами подготовки тестов.

Вероятностное тестирование. Функциональная схема организации показана на рис. 1.15, б и отличается наличием на входе ОД генератора псевдослучайных воздействий (ГПВ), реализованного сдвиговым регистром с обратными связями (РГС). Анализатор обрабатывает выходные реакции по определенным правилам (определяет математическое ожидание числа сигналов) и сравнивает полученные значения с эталонными. Эталонные значения рассчитываются заранее либо получают на предварительно отлаженном и проверенном таком же точно устройстве.

Компактное тестирование (сравнение с эталоном) заключается в том, что способ стимуляции может быть любой (генератор воздействий (ГВ) – ГПВ, программный), а эталонные реакции образуются в процессе тестирования с помощью дублирующего устройства (эталона). Анализатор производит сравнение выходной и эталонной реакций (рис. 1.15, *в*) [18].

При сигнатурном тестировании выходные реакции ОД, получаемые за фиксированный интервал времени, обрабатываются на регистре сдвига с обратными связями – сигнатурном анализаторе (СА) – и перекодируются (сжимаются) в короткие коды (сигнатуры). Полученные сигнатуры сравниваются с эталонными, которые получают расчетным путем либо на предварительно отлаженном устройстве. Стимуляция осуществляется с помощью ГВ или обычными программными методами, но только в фиксированном интервале времени, совпадающем с временем анализа (рис. 1.15, *г*).

Синдромное тестирование (метод подсчета числа переключения). Функциональная схема содержит ГТ, который генерирует два набора и подает их на вход схемы, а на ее выходе имеется счетчик, который производит подсчет числа единиц на выходе схемы. Если число единиц не равно эталонному значению, то схема считается неисправной.

Кольцевое тестирование показано на рис. 1.15, *д*. Устройство содержит универсальный регистр (Р), который может быть преобразован путем подачи двух управляющих сигналов в сигнатурный анализатор (СА) либо генератор псевдослучайных воздействий (ГПВ), которые поочередно подключаются к ОД через мультиплексор (МХ). В отличие от обычного СА, этот регистр образует сигнатуры сразу от всех выходных сигналов.

Поэлементное тестирование предполагает осуществление контакта со всеми выводами одной (или всех расположенных на плате) интегральной схемы, входящей в состав проверяемого устройства. При этом правильность функционирования каждой интегральной схемы определяется индивидуально (проверяемая интегральная схема "изолируется" от остальных, входящих в состав устройства). Контакт может осуществляться как со всеми элементами проверяемого устройства, так и индивидуальным подсоединением к каждой микросхеме [9].

Внутрисхемная эмуляция предполагает, что стимуляция проверяемых схем и анализ их реакций осуществляются через разъем проверяемого устройства.

Каждая из перечисленных организаций тестирования обладает разными требованиями к подготовке данных для процесса тестирования и обработке его результатов.

Рассмотрим более подробно основные принципы тестирования.

Программное тестирование для современных устройств требует предварительной автоматизированной подготовки тестовых воздействий и соответствующей обработки полученных реакций. Оно обладает гарантированной достоверностью обнаружения неисправностей заданного класса.

В соответствии с возможностью решения задач обнаружения и локализации можно выделить словарные и зондовые организации поиска неисправностей. Словарная организация, как правило, предусматривает автоматический режим диагностики. В простейшем случае смысл словарного поиска состоит в следующем. Определяется реакция устройства на выбранный набор тестовых сигналов. При проведении испытаний теми же тестами реального устройства реакция сравнивается с полученной информацией и по совпадению выполняется идентификация неисправностей. В непосредственном виде такой подход применяется редко, в основном в ЭВМ, поскольку хранение полных выходных векторов требует большой памяти (число

неисправностей и тестов велико). Для сокращения объема словарей используются методы сжатия двоичной информации.

Зондовая организация поиска (одно- и многократная) предусматривает полуавтоматический диалоговый режим проверки с участием инженера-оператора. Задача регулировщика заключается в подключении щупа к микросхемам, очередность подключения задается автоматически. Поиск неисправности осуществляется в статическом режиме. При подключении щупа к очередной микросхеме анализируется соответствие ее выходных реакций эталонным: если микросхема исправна, определяются питающие ее микросхемы, с которых поступает сигнал, отличный от требуемого.

Для реализации этого режима в конкретной системе необходимо иметь структурное описание схемы, библиотеку логических функций различных типов микросхем, входные и выходные эталонные тактовые сигналы и, кроме того, значения сигналов на всех полюсах исправной схемы при подаче данного исходного набора. Этот способ проверки используется как для всего блока, так и для отдельных его элементов. Поэтому различают поэлементный и поблочный контроль. Поэлементный контроль требует специального устройства – многоконтактного щупа.

Недостаток зондовой организации проверок заключается в необходимости обеспечить хороший контакт в месте соединения зонда и исследуемого элемента.

В связи с этим для современной аппаратуры, проектируемой на основе микропроцессоров, перспективным является использование бесконтактных зондов. В качестве такого зонда может использоваться прибор типа Щ4315-03, выпускаемый серийно. Использование этого прибора по определенной методике в сочетании с логическим анализатором позволяет значительно увеличить эффективность процесса тестирования и отладки сложных устройств на заключительных стадиях.

Подготовка тестов (определение набора входных воздействий и выходных реакций) для проверяемого устройства осуществляется физическим или математическим моделированием до проведения тестирования.

В первом случае в физическую модель (копию тестируемого устройства) вносятся неисправности и определяются входные воздействия, которые ее обнаруживают.

Во втором случае составляется машинная модель устройства и путем ее анализа определяют тесты. В связи со сложностью

современных устройств наиболее широкое распространение получил второй способ.

При реализации программного тестирования применяются условные и безусловные алгоритмы диагностирования и соответствующие им словарные и зондовые методы поиска дефектов.

Словарные методы применяются преимущественно в автоматическом режиме.

Применение словарей вызвано тем, что значительный объем результатов программного тестирования делает их непригодными для персонала, особенно когда необходимо по заданному симптому определить конкретную неисправность. Это обусловливает необходимость разработки специальных методов сжатия информации для получения удобной для ремонтников документации, называемой обычно словарем. Основным требованием, предъявляемым к словарям, является компактность представления.

Самый простой и неэкономичный тип словаря – словарь точного соответствия. Он представляет собой перегруппированные результаты моделирования исследуемого устройства на типовых воздействиях в присутствии неисправностей. Существуют два метода составления такого словаря: первый – с применением циклических кодов для более экономной упаковки данных моделирования и второй – с помощью фазовых микропрограмм.

Первый метод заключается в том, что признаки неисправности, представляющие собой многоразрядный двоичный код, перекодируются в более компактные малоразрядные признаки с помощью алгоритмов кодирования, применяемых для образования циклических кодов.

Второй метод состоит в том, что данные моделирования упорядочиваются по фазам прохождения микропрограмм.

Стремление сократить затраты и памяти для хранения словарей неисправностей в системах реального времени привело к тому, что в ряде случаев в качестве такого словаря используется сама программа моделирования, с которой проводятся эксперименты при различных, интересующих исследователя входных наборах.

Составление словаря неисправностей является итогом всей работы по составлению теста цифровой схемы. На практике эти словари имеют различные виды и формы: от подробных опи-

саний объемом в сотни страниц, до кратких таблиц, содержащих всего две колонки, в одной из которых помещаются в определенной последовательности входные слова, в другой – ассоциированные с ними неисправности.

Метод вероятностного тестирования основан на том факте, что всякое цифровое устройство, когда на его входы поступают случайные последовательности сигналов, выполняет вполне определенное, жестко связанное с его структурой преобразование распределения вероятностей этих сигналов. Соответствие указанного преобразования эталонному может при определенных условиях служить свидетельством правильности функционирования устройства в связи с тем, что снятие полной картины преобразования, осуществляемого цифровым устройством, затруднительно. Устройство объявляется исправным в том случае, если гипотеза о соответствии параметров случайных сигналов на его выходах некоторым эталонным значениям статистически подтверждается. Поэтому расчет данных для контроля заданной схемы сводится к определению границ областей принятия вышеуказанных гипотез и длины выборки значений сигнала на выходе схемы, которая обеспечила бы при проверке гипотез заданный уровень статистической достоверности.

Обычно параметром распределения выходных сигналов, относительно которого строится гипотеза, является математическое ожидание, а входные сигналы проверяемой схемы, генерируемые испытательной установкой, распределены по биноминальному закону и независимы между собой.

При этих предположениях для любого списка неисправностей комбинационной схемы может быть найден такой вектор вероятностей входных сигналов, что любые две неисправности этого списка (включая нулевую неисправность, соответствующую исправной схеме) порождают на выходе схемы сигналы с разными математическими ожиданиями и, следовательно, различаются статистическим методом.

Любая характеристика процесса контроля, в частности длина проверяющего эксперимента, может быть выражена как некоторая нелинейная функция от математических ожиданий входных сигналов схемы. Задача оптимальной по этому критерию организации процесса контроля может рассматриваться как задача нелинейного программирования.

Важной проблемой является также получение асимптотических оценок параметров вероятностного метода для сравнения их с соответствующими показателями других методов. Некоторые данные, полученные в этом направлении для комбинационных схем, показывают, что вероятностный метод по крайней мере не уступает тем из известных методов контроля, для которых подобные характеристики исследовались.

Перспективность вероятностного метода диагностирования для практического применения обусловлена в основном следующими двумя обстоятельствами. Во-первых, для него упрощается предварительный синтез тестовой последовательности. Во-вторых, метод позволяет при небольших аппаратных затратах обеспечить динамический режим работы схемы при проверке, а также дает возможность за счет высокой частоты аппаратного генерирования входной испытательной последовательности подать на схему большое число проверяющих наборов, что также повышает полноту обнаружения неисправностей. Недостатками метода являются ориентация в основном на комбинационные устройства, сложность при проверке устройств с памятью и отсутствие гарантированных оценок достоверности (полноты) контроля.

На практике случайные или псевдослучайные испытательные сигналы применяются в сочетании с сигналами, задаваемыми детерминированным способом.

Кольцевое тестирование и синдромное тестирование больше ориентированы на встроенную реализацию и менее распространены.

Методы **логического анализа** в настоящее время находят самое широкое применение для отладки и поиска дефектов как в аппаратуре, так и в программном обеспечении.

Этот метод является в какой-то мере аналогом ранее применявшегося в практике обслуживания ЭВМ метода измерений с помощью осциллографа. Его появление обусловлено тем, что для наладки современных цифровых систем часто необходимо видеть одновременно последовательности сигналов по нескольким каналам, наблюдать события, предшествующие интересующему нас событию, что невозможно при использовании аналоговых контрольно-измерительных приборов. Этот метод является специальным методом цифровых измерений, который позволяет инженеру "заглянуть в прошлое" и одновременно наблюдать сигналы по нескольким каналам.

Метод представляет в распоряжение инженеров технические средства – логические анализаторы, которые дают возможность проводить измерения и запоминать состояния устройства. Их использование сокращает время наладки примерно на 20%.

Метод поэлементного диагностирования заключается в проведении последовательности проверок каждого компонента или фрагмента схемы печатных узлов с учетом обеспечения условий исключения взаимного влияния электрорадиоэлементов (ЭРЭ) и неповреждающего характера тестовых воздействий. Метод поэлементного диагностирования не заменяет, но существенно дополняет метод обычного тестирования узлов радиоэлектронной аппаратуры (РЭА). Если применять его и системы тестирования в одном технологическом процессе изготовления РЭА, то они, дополняя друг друга, позволяют увеличить выпуск и качество продукции, создают экономию капитальных вложений и трудовых затрат.

Основным отличительным признаком структуры является наличие специального игольчатого контактного устройства (ИКУ), предназначенного для подачи поэлементных тестовых воздействий и снятия соответствующих реакций.

Объектом поэлементного диагностирования (ОД) являются аналоговые, цифровые и гибридные печатные узлы (ПУ) РЭА. Как правило, система поэлементного диагностирования (СПД) обнаруживает следующие виды дефектов в ПУ: короткие замыкания, обрывы, неправильно ориентированные ЭРЭ, выход параметра ЭРЭ за пределы допуска, монтаж ЭРЭ другого типа, неправильное функционирование элемента или фрагмента схемы (системы, в которых предусмотрена возможность тестирования как отдельных ЭРЭ, так и фрагментов ПУ, называют также системами покомпонентного диагностирования).

Диагностирование ОД осуществляется в следующей последовательности (с целью обеспечения неповреждающего характера проверок и скорейшего обнаружения и устранения производственных дефектов):

контроль целостности проводников,

обнаружение ложных соединений,

проверка ориентации ЭРЭ,

контроль выбранных параметров ЭРЭ на микрорежимах,

тестовая проверка ЭРЭ и подсхем при номинальном напряжении питания,

функциональные испытания узла в целом.

Метод внутрисхемной эмуляции ориентирован на использование в процессе отладки и поиска неисправностей в микропроцессорных системах. Функции и режим работы МПС сохраняются почти такими же, как в реальном применении, обеспечивается программно-управляемый доступ к схемам МПС для задания различных отладочных и диагностических режимов работы, внесения изменений, оценки состояния.

1.6. МЕТОДЫ ПРОВЕРКИ И ОТЛАДКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Использование логических анализаторов для отладки программного обеспечения.

Расширение области применения и усложнение комплекса функциональных и эксплуатационных возможностей микропроцессорных систем во многом определяются эффективностью их программного обеспечения.

При разработке новых МС стоимость их ПО достигает 60% всех затрат на проектирование. Однако качественные показатели ПО современных МС в настоящее время совершенствуются гораздо медленнее, чем технико-экономические характеристики аппаратных средств. Поэтому при отладке ПО возрастает значение фактора эффективного использования современных средств и методов контроля ПО. Одним из таких средств является логический анализатор (ЛА). По сравнению с известными системами разработки МС, ЛА отличается более низкой стоимостью работы в реальном времени.

Применение ЛА для проверки ПО современных МС позволяет проводить оптимизацию программ, сравнительную оценку идентичных программ, обнаруживать ошибки в ПО.

Отыскивание ошибок в ПО осуществляется путем контроля операторов в кодах МП в процессе последовательной проверки хода выполнения программы. Логический анализатор позволяет реализовать пультовые режимы работы при отладке рабочей программы в диалоговом режиме. Возможно пошаговое выполнение инструкций, остановы по заданным условиям, контроль последовательности вызова подпрограмм, процедур обмена между устройствами МС, вводимой внешней информации, а также вывод на экран значений переменных и массивов.

Оптимизация программ и сравнительная оценка идентичных программ проводятся с помощью количественных показателей.

В качестве показателей используются: время выполнения, число обращений к подпрограммам, емкость и область использования памяти, число инструкций.

Возможность выполнения перечисленных проверок обеспечивается благодаря весьма развитому набору систем запуска регистрации ЛА.

Использование ЛА для отладки ПО МС позволяет сокращать время отладки, определять "узкие" места в ПО, связанные с медленным выполнением, создавать программы с минимальным объемом и временем исполнения.

Подробно методические вопросы изложены в гл. 7.

Глава 2

МАТЕМАТИЧЕСКИЕ МОДЕЛИ ДЛЯ ОПИСАНИЯ ПРОЦЕССОВ И ОБЪЕКТОВ ОТЛАДКИ В СОСТАВЕ МИКРОУВК

2.1. АЛГОРИТМИЧЕСКИЕ ОПИСАНИЯ ПРОЦЕССА ОТЛАДКИ И ЭЛЕМЕНТАРНЫЕ ДИАГНОСТИЧЕСКИЕ ОПЕРАЦИИ

Как отмечалось, в процессе отладки взаимодействуют объект отладки, средства отладки и внешняя среда. В процессе отладки осуществляется большое число подготовительных и отладочных операций. Формальное описание взаимодействия этих объектов позволяет выявить узкие места и выбрать наиболее эффективные решения, особенно для таких сложных объектов, как МПС [37, 38].

В общем случае процесс взаимодействия может быть описан алгебраическими либо графовыми моделями. К алгебраическим относят микропрограммные алгебры, к графовым – сети Петри и др. [39].

Рассмотрим некоторые из таких описаний. Вначале остановимся на алгебраическом описании, осуществляемом в терминах элементарных диагностических операций и являющимся разновидностью микропрограммных описаний.

Введем понятие *элементарной диагностической операции* (ЭДО), под которой будем понимать элементарное действие f_i^τ, $i = \overline{1,k}$; реализуемое за время τ программными или аппаратными средствами в составе более крупных отладочных опера-

ций. К диагностическим операциям относятся: формирование стимулирующих воздействий, подача их на объект проверки, снятие выходных реакций, сравнение их с эталонными, принятие решения по результату сравнения, переход к следующей проверке и др.

Обозначим ЭДО через f_i^τ, каждой ЭДО можно поставить в соответствие характеристики качества R_{ij}, $j = \overline{1,n}$. Простейшими операциями могут быть следующие: обнаружение $f_о^\tau$, локализация $f_л^\tau$ и исправление $f_и^\tau$ ошибки. Каждая из операций состоит из ряда элементарных операций, например, обнаружение состоит в подаче входного воздействия, съеме выходной реакции, сжатии ее и сравнении с эталоном, который хранится в накопителе либо формируется специальным образом, т.е.

$$f_о^\tau = \{ \theta_п^\tau(x), \theta_с^\tau(y), \theta_s^\tau(r), \theta_\oplus^\tau(r_э, r)\}.$$

В свою очередь, $\theta_\oplus^\tau$ может включать в себя такие ЭДО: $\pi_\oplus^\tau$ – суммирование по модулю 2; π_Σ^τ – принятие решения и др.

Представление процесса диагностирования в виде совокупности ЭДО является полезным для поиска новых способов и сочетаний проверок. Каждая из ЭДО реализуется различными средствами и с помощью различных принципов.

Параллельную реализацию совокупности ЭДО запишем в виде

$$F^{пр} = \left| \begin{matrix} f_1 \\ \vdots \\ f_k \end{matrix} \right|_{i=1}^{k};$$

последовательную реализацию – следующим образом:

$$F^{пл} = \left| \; f_1, f_2, \; . \; . \; . \; , f_k \; \right|_{i=1}^{k}.$$

Число тактов преобразования диагностической информации

обозначим z, реализуемую ЭДО – f_z^{τ}, а условную стоимость реализации ЭДО – $C(f, z)$ (для конкретных систем зависит от z в значительной степени).

Структурно-временная реализация ЭДО. Основные требования к диагностическим операциям состоят в следующем.

Следует обеспечить высокую скорость подачи входных воздействий и съем выходных реакций для проведения проверок не только в статических, но и динамических режимах.

Диагностические операции должны обеспечивать сжатую форму информации и не требовать для своей реализации значительных затрат памяти, времени, аппаратуры.

Рассмотрим один из способов поиска новых типов ЭДО и их новых структурно-временных реализаций, который представляет интерес для построения эффективных средств диагностирования.

Операция сравнения для контроля проводится применительно к отдельным кодовым словам x, y, их сжатым представлениям r_x, r_y; к массивам слов M_x, M_y, что запишем так:

$$f_{\oplus}^{\tau 1}(x, y), f_{\oplus}^{\tau 2}(r_x, r_y), f_{\oplus}^{\tau 3}(M_x, M_y).$$

Операция сравнения, как правило, реализуется путем суммирования по модулю 2: $f_{\oplus}(x, y) = x \oplus y = \bar{x}y \vee x\bar{y}$.

Известны более сложные реализации этой операции, например, путем сравнения полученного значения y с допусками Δ (верхним и нижним пределами): $y - \Delta < y < y + \Delta$.

Рассмотрим другой класс операций сравнения, выполняемых с помощью операций max и min и использующих динамические свойства исследуемых объектов.

Введем в рассмотрение длительность выполнения группы основных операций $\mu_о$ и эталонную длительность $\mu_э$. Операцию сравнения, состоящую в проверке условия $\mu_о = \mu_э$, реализуем с помощью операций max и min. Вид операций зависит от типа ошибки $\mu^{\varepsilon} > \mu_э$, $\mu^{\varepsilon} < \mu_э$ и применяемых физических принципов представления кодов в устройствах; операцию сравнения представим в виде

$$f_{\oplus}^{\tau 4} = \max{}^{\tau}(\mu_о, \mu_э).$$

Здесь τ – момент времени, в который происходит сравнение. Операции max и min реализуются логическими функциями И, ИЛИ, обладающими экстремальными свойствами. При этом эталон μ является виртуальным. Реализация контроля с использованием экстремальных свойств указанных функций предполагает выполнение дополнительных операций, таких, как формирование начала $\varphi_н$ и конца $\varphi_к$ временно́го интервала, "окна" $T_и$, на котором выполняется сравнение, формирование сигнала стробирования $\varphi_с$.

Признаками для формирования интервала $T_и$ могут служить различные состояния диагностируемого объекта, что дает возможность строить широкий класс проверяющих устройств.

С использованием ЭДО алгоритм функционирования сигнатурного анализатора совместно с объектом диагностирования упрощенно описывается набором операторов $A = \{\varphi_н, \varphi_к, \varphi_с\}$. Изменение кода сигнатуры эквивалентно нарушению длительности временно́го интервала, определяемого "окном" анализатора.

Аналогично может быть описано формирование контрольных операций в логическом анализаторе, в котором есть широкий спектр микроопераций регистрации и отображения информации о временны́х состояниях в различных точках дискретного устройства. Это дает возможность применять операции логического и сигнатурного анализа как для отдельных устройств, так и для системы в целом, в том числе во встроенном варианте, при этом применять не только логические, но и логико-динамические признаки обнаружения ошибок.

Логико-динамические признаки. Признаком правильности функционирования МПК будем считать пару $\{\mu_i = \mu_i^э, R_i = R_i^э\}$, $i = \overline{1,n}$, где μ_i – длительность выполнения основной операции; $\mu_i^э$ – эталонное значение; R_i, $R_i^э$ – получаемый и эталонный результаты.

Таким образом, процесс диагностирования можно представить в виде совокупности операторов φ_i, $i = \overline{1,k}$, обеспечивающих выполнение этого условия. Например, с учетом этого условия для микропроцессора методика предварительной подготовки проверок сводится к следующему:

1) составляют динамическую модель $\widetilde{\mathbf{M}}$, учитывающую устройства, связи между ними и задержки;

2) проводят анализ динамической модели $\tilde{\mathbf{M}}$ и получают длительности μ_z путей L_z прохождения информации ($z = \overline{1,m}$);

3) подсчитывают число синхросигналов для получения μ_z;

4) определяют длительностями μ_z "окно" в сигнатурном анализаторе;

5) вводят в сигнатурном анализаторе несколько "окон" $T_и(\mu_z)$.

При проверке эталонные сигнатуры для μ_z должны совпадать с полученными. Наличие этой методики повышает достоверность диагностирования в результате охвата контролем большего объема микропроцессора.

Аналогичная методика применяется для других устройств МПК и на уровне системы в целом.

Таким образом, временные диагностические операции позволяют рассматривать МПК как динамический объект и более адекватно описывать процесс его взаимодействия со средствами проверки и наладки.

Предложенные описания можно использовать при построении ЛА и СА и при организации проверок, причем могут быть построены компактные алгоритмы как укрупненные, так и подробные.

Дальнейшей формализацией является построение специального языка для автоматизации составления сложных программ проверок. Например, в рамках введенного описания работа ЛА описывается выражениями вида

$$y_i[A_j * R_i \oplus G_q \oplus Y_p],$$

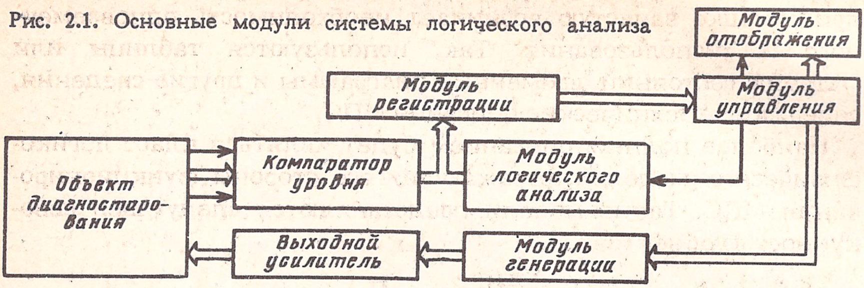
где A_j – оператор анализа, $j = \overline{1,n}$; R_i – оператор регистрации, $i = \overline{1,m}$; G_q оператор генерации, $q = \overline{1,l}$; Y_p – участок информационного потока, на котором выполняются операторы, $p = \overline{1,s}$; * – знак операции следования или логической операции.

К операциям следования относятся высказывания "Если", "То", "Перейти", "Печать", "Окончить". Логические операции: дизъюнкция, конъюнкция.

На рис. 2.1 приведена схема модулей логического анализа, реализующая это выражение. Она включает модули регистрации, логического анализа, отображения.

Вводя на основе этих описаний различные конструкции, например конструкции операторов, конструкции условий, элементарные конструкции типа числа, идентификатор, раз-

Рис. 2.1. Основные модули системы логического анализа



мерность, получают формализованный язык, который применяют для составления программ проверок.

Например, запись 12 ТАКТ ВОЗДЕЙСТВИЕ "1010" означает – генерировать входное воздействие 1010 в момент появления синхронизации, после 12 синхроимпульсов окончить генерацию.

Запись КВАЛИФИКАЦИЯ 01X10 10 ТАКТ означает, что установленные условия выполнены, когда в информационном потоке появилась информация вида 01X10 и после этого прошло 10 синхроимпульсов.

2.2. МАТЕМАТИЧЕСКИЕ МОДЕЛИ ДЛЯ ОПИСАНИЯ МПС И СОСТАВЛЕНИЯ ОТЛАДОЧНЫХ ТЕСТОВ И ПРОЦЕДУР

При отладке, также как и для других этапов жизненного цикла МПС, возникает проблема выбора математического описания объектов наладки и диагностирования. Такое описание позволяет алгоритмизировать процессы подготовки и отладки, обеспечивает переход к автоматизированным методам и таким образом способствует общему сокращению времени на эти процессы. Практически это означает возможность получения эталонных значений и тестов. Математические модели должны отражать различные аспекты исследуемого объекта с достаточной степенью точности и достоверности и учитывать реальные особенности МПС (такие, как, например, параллельное протекание процессов, наличие программных и аппаратных компонентов), отражать не только статические, но и динамические процессы.

В отличие от этапов проектирования, где имеется определенная последовательность использования тех или иных математических моделей и алгоритмических описаний во времени,

при отладке зачастую возникает необходимость одновременного их использования. Так, используются таблицы или матрицы состояний, временны́е диаграммы и другие сведения, содержащие технические описания МПС.

Наиболее полным описанием будет являться класс логико-физических моделей, отражающих все стороны функционирования МПС. Такие модели представляются следующей совокупностью объектов:

$$M = \langle Л_1, Л_2, \ldots, Л_k, П_1, П_2, \ldots, П_z \rangle,$$

где $Л_1, \ldots, Л_k$ – параметры логических моделей; $П_1, \ldots, П_z$ – параметры физических (электрических) моделей. Модели такого типа предназначены для установления причинно-следственных связей и отношений между объектами и их поведением в составе МПС.

В общем случае при пользовании такими моделями может получаться качественная (не количественная) информация, т.е. может существовать несколько ответов на вопросы, возникающие в процессе отладки. Модели такого типа, называемые "мягкими", перспективны для систем, обладающих функциями искусственного интеллекта.

На настоящий момент времени используют сочетание независимых моделей, отражающих те или иные стороны МПС. К ним относятся модели логического типа – поведенческие и структурные, модели электрического уровня, топологические.

В свою очередь, в рамках каждой из моделей могут быть использованы известные математические описания – алгебраические и теоретико-графовые.

Для представления на программном и микропрограммном уровнях логического описания может быть использовано представление МПС в виде композиции двух устройств – операционного (ОУ) и управляющего (УУ).

Операционное устройство определяется следующей совокупностью объектов:

$$ОУ = \langle Ф, W, U, |L|, O \rangle,$$

где Ф – множество функциональных объектов; W – множество ячеек памяти; $|L|$ – матрица связей между блоками; O – множество микроопераций, определенных на Ф и W.

Управляющее устройство представляется следующей совокупностью объектов:

$$УУ = \langle I, \; Ф, \; U, W \rangle,$$

где I – множество микрокоманд.

Такое представление является полезным не только для представления МПС или процессора в целом, но и для отдельных устройств и блоков, входящих в ее состав.

Структурные модели отражают в основном структурные свойства, т.е. блоки и связи между ними.

Обобщенная форма описания отлаживаемого объекта может быть представлена в виде графа

$$G(E, U),$$

где E – вектор, $E = \{ e_0, R_1, R_2, \ldots, R_s \}$; e_0 – функциональный оператор, реализуемый аппаратно или программно; $R_1, R_2, \ldots, R_s$ – параметры выполнения (временны́е операторы, структурные и функциональные); U – связи между операторами.

Таким образом, полезным является рассмотрение МПС как логико-динамической системы, в которой признаками правильности функционирования являются функциональные, структурные и временны́е соотношения.

С помощью моделей такого типа определяют ошибки синтеза и ошибки исходного задания алгоритмов (обычно их доля до 40%). Модели служат для получения эталонов, с которыми необходимо сравнивать реальные значения сигналов отлаживаемого объекта в определенных точках.

Обнаружение этих ошибок осуществляется с помощью структурного анализа и контроля всех операций функционального преобразования информации.

При структурном анализе проверяются занесение и считывание информации для всех регистров через все информационные пути.

Функциональный контроль производится для всех операций преобразования информации с целью комплексной проверки всего набора операций и их алгоритмов на соответствие техническим требованиям.

Как отмечалось, в процессе отладки важен не только выбор стратегий и размещения контрольных точек по структуре МПС, но и подготовка тестовых воздействий, для чего можно ис-

пользовать как рассмотренные выше модели, так и программные средства их реализации. Наиболее перспективно использование языков высокого уровня для написания тестов, обеспечивающих минимальную трудоемкость создания тестовых программ большого объема, диалоговые режимы работы, легкость обучения и возможность работы персонала средней квалификации, минимальные затраты времени и средств на исправление ошибок и внесение изменений, удобство проведения эксперимента.

Ниже рассмотрим модели, которые наиболее часто используются в практических исследованиях.

Наиболее распространены графовые модели, обеспечивающие достаточно ясное отображение структурных и динамических свойств объектов различной физической природы (программ и аппаратуры), отображение параллельных процессов, управления и обработки информации.

К числу графовых моделей относятся сети Петри [40], граф-схемы алгоритма (ГСА). Их достоинство в универсальности, единообразии форм представления и наглядности, возможности разработки достаточно эффективных алгоритмов анализа. Недостатком является сложность отображения объектов большой размерности.

К алгебраическим моделям относятся микропрограммные алгебры, логические схемы алгоритма (ЛСА) и другие, которые обладают возможностью компактного представления устройств большой сложности, наличием эффективных средств анализа и преобразования.

Диагностическая модель МПС на основе сетей Петри содержит два компонента:

сеть Петри PN:

$$PN = \{ P, T, U \},$$

где P – множество позиций, $P = \{ p_1, p_2, \ldots, p_n \}$; T – множество переходов, $T = \{t_1, t_2, \ldots, t_m \}$; U – множество направленных дуг (сеть Петри моделирует процессы управления в МПС);

и граф данных GD:

$$GD = \{ E, Z \},$$

где Z – множество направленных дуг; E – множество вершин

графа, $E = \{W_i, Q_j, O_k\}$. Здесь W_i – множество ячеек памяти; Q_j – множество операторов; O_k – множество ячеек предиката.

Дуги графа задаются следующим образом: $W(O_j)$ – множество ячеек памяти, из которых происходит чтение при выполнении оператора O_j; $R(O_j)$ – множество ячеек памяти, в которые происходит запись при выполнении этого же оператора, тогда

(W_i, O_j) – дуга, если $W_i \in W(O_j)$;

(R_i, O_j) – дуга, если $W_i \in R(O_j)$;

(W_i, O_k) – дуга, если $W_i \in W(O_j)$.

Взаимодействие между компонентами диагностической модели достигается путем простановки у переходов сети Петри дополнительных условий срабатывания и списка операций, выполняемых при срабатывании перехода. Каждый переход t_m сети помечается парой (g_l, O_l), где g_l – логическая функция предиката Q_k; O_l – список операторов, определенный в графе данных.

Если g_l – ложно, то переход не срабатывает, если g_l – истинно, то переход t_m разрешен и срабатывает.

Подготовка стимулирующих тестов для модулей МПС при логическом анализе. Подготовка тестов остается важной задачей, так как позволяет обеспечить высокий уровень достоверности проверок при достаточно малых затратах памяти. Задача поиска упрощается, так как необходимо определять неисправность с точностью до неисправной ИС. Обычно топология микропроцессорных модулей неизвестна, поскольку составляет секрет разработки, а необходимо ориентироваться на исходную информацию в виде различного ряда функциональных описаний, например набора команд, микрокоманд и пр.

Для построения тестов микропроцессоров полезной оказывается последовательная идентификация моделей и оперирование классами неисправностей. Этот подход предполагает введение и применение иерархии агрегированных моделей: модели на уровне программных модулей и аппаратных блоков, на уровне отдельных команд и микрокоманд.

Пусть M_j^0 – модель j-го уровня $(j = \overline{1, k})$, отвечающая исправному объекту, M_j^ε – модель с неисправностью. Задача иденти-

фикации состоит в нахождении входных воздействий и выходных реакций, определяющих M_j°. Если при проверке подтверждается, что M_j° существует, то можно считать процесс диагностирования j-го уровня законченным. В противном случае испытания продолжаются в классе моделей M_j^{ε}.

При таком подходе используют последовательные процедуры проверки и соответствующие им классы моделей дискретных систем и иерархическую последовательность тестов.

Если система полностью исправна, то результатом диагностирования будет последовательность идентифицированных в ходе диагностирования моделей $M_1^{\circ}, M_2^{\circ}, \ldots, M_k^{\circ}$ (где индекс k означает уровень модели).

Отсутствие модели M_j° для любого уровня свидетельствует о наличии ошибки. Такая процедура ориентирована на последовательное обнаружение различных типов ошибок (монтажа, проектировщика и др.).

Процесс идентификации первого уровня даже при наличии неисправностей осуществляется более просто и быстро. Ему можно поставить в соответствие словарь неисправностей первого уровня, по которому проводится локализация ошибок первого уровня.

При нахождении модели M_j° осуществляется идентификация моделей второго уровня, затем следующего уровня. На каждом уровне оперируют с сокращенными описаниями и соответствующими подмножествами неисправностей. Такая процедура приводит к построению словаря с иерархической структурой. В отличие от обычных процедур условного (последовательного) поиска идентификация проводится для всей иерархии моделей последовательно, что значительно сокращает процедуру перебора. Использование этого подхода требует формулировки условий идентификации. Идентифицируются команды, микрокоманды, пути в графовом представлении микропрограмм (цепочки микрокоманд). В общем виде условие идентификации записывается следующим образом:

$$Q(X, Y, t, \delta) = Q(X, Y_э, t, \delta),$$

где δ – параметр, подлежающий идентификации; X – входное воздействие; Y – выходная реакция на воздействие X.

Для микропрограммного уровня это условие имеет вид

$$Q(m_1, m_2, x_1, x_2, y) = Q(m_1, m_2, x_1, x_2, y_э),$$

где m_1, m_2 – микрокоманды; x_1, x_2 – операнды; y – результат; $y_э$ – эталон.

Идентификация путей осуществляется подбором как простых операндов, так и операндов аддитивно-отличающихся чисел, любые суммы которых различаются. К ним относятся последовательности вида 2^r, $r = \overline{1, n}$ и др.

2.3. МОДЕЛИ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

В вычислительном процессе (ВП) реализуются в реальном времени все компоненты программного обеспечения. Рассмотрим модели ВП, пригодные для решения задач диагностирования и отладки.

Различие моделей организации ВП в рассматриваемом случае основано на различных подходах в решении задач диагностирования ПО и на различной функциональной ориентации задач, решаемых на базе той или иной модели ВП. Отличие предлагаемых моделей от известных или традиционных подходов в организации ВП состоит в том, что рассматриваемые модели ориентированы для решения задач диагностики ошибок в функциональной части ПО.

Первая из таких моделей функционально ориентирована на описание выполняемых задач управления технологическим объектом. Основой формального описания модели является диаграммный граф [46, 48, 49]

$$G_D = \langle V, P, T \rangle,$$

где V – множество вершин; P – множество ребер; T – множество точек синхронизации.

Диаграммный граф – это средство наглядного представления и описания процессов в задачах управления. В G_D легко формировать условия проверки отсутствия ошибок, которые могут проявиться в процессе решения задачи. Методы оперативного контроля и восстановления, вытекающие непосредственно из

модели G_D, представляют собой алгоритмы диагностирования (которые связаны с алгоритмом функционирования всей модели диаграммного графа) и алгоритмы анализа средств программного отображения модели, содержащие данные конкретной решаемой задачи. Алгоритм функционирования G_D представляется в виде неформального описания правил, регламентирующих передачу управления с одной точки G_D в другую. Основными элементами, по отношению к которым описывается алгоритм передачи управления, являются точки синхронизации с нижними и верхними индексами c_i^j и линии синхронизации c_i. В G_D для индексов точек c_i^j должны выполняться определенные отношения между нумерацией индексов, которые являются критериями диагностирования ошибок. Поэтому алгоритмы диагностирования на каждом шаге работы алгоритма функционирования G_D осуществляют проверку выполнимости требуемых соотношений или ограничений и в случае их нарушений фиксируется факт ошибки. Алгоритмы диагностирования ошибок не ограничиваются только рамками модели G_D, а работают с программными средствами отображения компонентов и модели в целом. Это позволяет не только обнаруживать ошибку, но и определять ее тип или класс в рамках принятого в [45] подхода к классификации ошибок. Эта возможность позволяет строить также эффективные средства корректировки диагностируемых ошибок. Например, одним из атрибутов программного отображения G_D является таблица логики реализации модели T_L. Эта таблица содержит двоичное отображение последовательности логических команд ветвления, номера команд r из таблицы команд используемого языка программирования и адреса меток используемых вершин α. Одним из алгоритмов поиска логической ошибки является следующий алгоритм:

1. Проверяется, равна ли длина l количеству команд в анализируемой ветви.

2. Если равна, то ошибка категории – неправильный логический анализ. Переходят к п. 4.

3. Если длина l больше количества команд ветви, то ошибка категории – пропуск логического анализа. Переходят к п. 5.

4. Проверяется очередная команда логического анализа функции l ветви на соответствие элементу в таблице T_l. Если

соответствие есть, то переходят к следующей команде, если его нет, то производят необходимую модификацию анализируемой команды, используя информацию из соответствующего элемента r таблицы T_L.

5. Проверяется очередная команда в ветви, если она не соответствует своему элементу в l, проверяют ее соответствие следующему элементу из l. Если соответствие появляется, то возвращают к предыдущему элементу из l и по соответствующему значению элементов из r и α таблицы T_L определяется пропущенная команда. Производится раздвижение ветви на необходимое количество байт и вставляется соответствующая команда.

6. Переходят к очередной команде ветви.

Приведенный в качестве примера алгоритм диагностирования работает в рамках следующих ограничений:

предполагается, что в ветви существует только одна ошибка категории пропуска команды при последовательном рассмотрении команд ветви;

каждой команде логического анализа и передачи управления на основе этого анализа должна предшествовать команда, которая непосредственно осуществляет при соответствующих условиях такое изменение или является последней командой цепочки команд, осуществляющих такое изменение в определенных ситуациях.

Математические модели отдельных программ. Технологические отладки отдельных программ состоят из нескольких этапов. На первом этапе без исполнения программ осуществляется ее проверка на соответствие некоторым формальным правилам или дополнительно выданным спецификациям (этот этап называют еще верификацией). На втором этапе (обычно называемом тестированием) программа проверяется на некоторых исходных данных, называемых тестовыми. При этом решаются такие задачи проверки, как правильность следования операторов, контроль распределения памяти и др.

При решении возникающих задач могут использоваться алгебраические и графовые диагностические модели.

Алгебраические модели используются для верификации, доказательства функциональной корректности, законченности программ. Теоретико-графические методы применяют в большей степени для предварительного структурного анализа

программ (без исполнения) и для проверки и локализации ошибок в процессе выполнения программ.

Теоретической основой представления программ являются операторные схемы программ.

В настоящее время используется представление программы в виде взвешенного графа

$$G_V = \langle V, X, \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4 \rangle,$$

где V – множество вершин графа, сооответствующих участкам команд программы; X – множество вершин, которые соответствуют исходным данным, константам и переменным (V и X не пересекаются); Γ_1 – отображение на графе выражает отношение следования по управлению между отдельными частями программы (совокупность $\langle V, \Gamma_1 \rangle$ определяет структурную схему программ); Γ_2 отражает отношение принадлежности величин к числу исходных данных отдельных частей программ, отношение к числу результатов, полученных в процессе вычисления.

Совокупность $\langle V, X, \Gamma_1, \Gamma_3 \rangle$ определяет схему функциональных связей программы, которая соответствует функциональной граф-схеме.

Информационные связи между величинами определяются отображениями Γ_4 множества X.

Таким образом, совокупность $\langle V, X, \Gamma_1, \Gamma_2, \Gamma_3 \rangle$ определяет структурные, функциональные и информационные связи программы. Вершины и дуги графа могут взвешиваться компонентами, имеющими числовые значения: количество команд в участке программы, количество ячеек памяти, время выполнения участка, вероятность выполнения.

Для отладки могут использоваться такие отдельные совокупности:

$$\langle V, X, \Gamma_1 \rangle, \langle V, X, \Gamma_2 \rangle, \langle V, X, \Gamma_3 \rangle, \langle V, X, \Gamma_4 \rangle.$$

Конкретные технологические задачи сводятся затем к решению ряда задач на графах указанного вида.

2.4. ВОПРОСЫ РАЦИОНАЛЬНОЙ ОРГАНИЗАЦИИ ПРОЦЕССА ОТЛАДКИ И ДИАГНОСТИРОВАНИЯ

Технологически процесс отладки состоит, как отмечалось, из ряда операций, выполняемых техническими средствами и человеком-оператором. Рациональная организация процесса отладки позволяет сократить время и затраты, повысить производительность труда в процессе отладки и проверки. Организация процесса отладки предполагает выбор общих принципов проведения процесса отладки, выбор комплекса технических (аппаратных и программных) средств, выбор точек их подключения к объекту наладки, выбор стратегий и способов их взаимодействия в процессе отладки и диагностирования. Количественно процесс отладки характеризуется временем $T_н$ и стоимостью процесса $C_н$.

Полное время наладки в общем случае состоит из следующих составляющих:

$$T_н = T_п + T_а + T_к,$$

где $T_п$ – время подготовки тестовых воздействий; $T_а$ – время автономной отладки; $T_к$ – время комплексной отладки.

Стоимость $C_н$ определяется стоимостью технических средств, квалификацией персонала и другими компонентами.

Рассмотрим отдельные вопросы рациональной организации отладки. В общем случае из теоретических соображений процесс отладки можно представить как последовательные процедуры декомпозиции отлаживаемого устройства по структуре и выполняемым функциям с последующей идентификацией каждого участка. Для описания процессов наладки наиболее подходят последовательные методы.

Последовательные методы могут быть реализованы: снизу – вверх, сверху – вниз, а также комбинированно.

Реализация их связана с фундаментальными понятиями управляемости и наблюдаемости диагностических систем. Будем рассматривать объект наладки как динамическую систему $S(X, t)$, которая называется полностью наблюдаемой, если для любого времени t_0 можно косвенно или непосредственно по результатам измерения вектора наблюдения $Y(t_0)$ определить вектор состояния $X(t_0)$ на конечном интервале времени t_0, t. Полная наблюдаемость имеет место, если $Y(t) = \alpha X(t)$, где $\alpha = \text{const}$. Система называется полностью наблю-

даемой в некоторой области D, если из некоторого начального состояния $Y(t_н) \in D$ она может быть переведена в любое другое состояние $Y(t_1) \in D$ за конечный интервал времени $\tau = t_1 - t_н$ воздействием на нее управления $X(t) \in G$ (G – некоторый класс). Система называется управляемой не полностью, если имеется такая подобласть $D' \in D$, где не существует уравнение, с помощью которого можно было бы перевести систему в состояние $Y(t_1) \in D'$. Для различных классов систем существуют соответствующие критерии.

По существу отладка и диагностирование представляют собой реализацию условий управляемости и наблюдаемости для системы $S(x, t)$, функционирующей в условиях неисправностей.

Введем понятие стоимости реализации наблюдаемости $C_{наб}$ и стоимости реализации условий управляемости $C_у$. Для сложных дискретных систем в целом, какой является МПС, значения $C_{наб}$ и $C_у$ оказываются высокими. Поэтому используют декомпозицию по структуре и функциям и последовательные процедуры проверки, что обеспечивает приемлемые затраты времени и стоимость.

Таким образом, возникает задача разделения МПС на отдельные модули, выбор средств для отладки и способов их взаимодействия.

В общем случае выбор организации основывается на предварительном анализе структуры МПС и различных организаций процесса тестирования и отладки, т.е. выбора способов подготовки и подачи стимулирующих воздействий, способов съема и анализа выходных акций.

Так как в настоящей работе сделана ориентация на методы сигнатурного и логического анализа, то в дальнейшем будем использовать это обстоятельство. Выбор точек подключения логического анализатора (ЛА), точек синхронизации, подготовка стимулирующих воздействий, разработка сценариев проверки составляет организацию процесса наладки на основе ЛА.

Пусть проверяемая система S разделена предварительно на модули $S_1, S_2, \ldots, S_k$. Пусть имеется набор диагностических операций $O = \{o_1, o_2, \ldots, o_l\}$, которые необходимо выполнить, и набор $X = \{x_1, x_2, \ldots, x_k\}$ технических средств, с помощью которых они реализуются.

Для каждой операции $o_i \in O$ введем показатели качества – достоверность $R_i(x_i)$, время проверки $\tau_i(x_i)$, стоимость $l_i(x_i)$.

Тогда задача выбора рационального состава средств проверки может быть сформулирована так: найти такое $x_0 \in X$, чтобы обеспечить

$$\max_{x_0 \in X} R(X)$$

при

$$\tau(X) \leqslant \tilde{\tau},$$

$$C(X) \leqslant \tilde{C},$$

где $\tilde{\tau}$, $\tilde{C}$ – вариант значений τ и C. Эта задача решается с применением известных методов решения экстремальных задач на графах либо методом динамического программирования.

Вариант технических средств X_j формируется на основе построения морфологических блоков. Морфологическим блоком средств проверки называется матрица, в которой столбец соответствует различным типам средств реализации элементарных операций, а строка – различным вариантам средств проверки одного типа.

Вариант $X_j = \{x_1, x_2, \ldots, x_k\}$ формируют таким образом: из каждого морфологического блока, принадлежащего модулю S_i, выбирают по одному элементу x_i, составляют вектор X_j, который используют в дальнейшем.

Формирование вариантов X_j для модулей S_i позволяет выбирать и оценивать различные сочетания стимулирующих и решающих операций. Кроме того, можно подбирать под типы модулей и решающие задачи соответствующие типы ЛА – программные микропроцессорные.

В случае необходимости могут быть сформированы требования к техническим характеристикам используемых ЛА, а также требования к их разработке.

В качестве таких характеристик могут использоваться быстродействие, число каналов, емкость памяти по каждому каналу, стоимость, масса.

Наряду с количественными характеристиками при выборе типов ЛА используются и качественные – удобство работы с ЛА, подготовка исходных данных, подключения, отображение информации в удобном виде и др.

Как представляется, процедура выбора набора ЛА должна носить итерационный характер и использовать сочетание формальных и качественных методов отбора типов устройств.

Глава 3

ВЫБОР СТРУКТУРЫ СИГНАТУРНЫХ АНАЛИЗАТОРОВ, ПРЕДНАЗНАЧЕННЫХ ДЛЯ ДИАГНОСТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ КОМПЛЕКСОВ, И РАСЧЕТ ОСНОВНЫХ ПАРАМЕТРОВ

3.1. ВЫБОР СТРУКТУРЫ СИГНАТУРНОГО АНАЛИЗАТОРА ДЛЯ ПОИСКА НЕИСПРАВНОСТИ В МПК

Трудность поиска неисправности в цифровой схеме заключается в том, что для некоторых типов отказов сложно отличить на экране осциллографа осциллограмму правильно работающего выхода схемы от осциллограммы отказавшего. Поэтому требуется глубокий анализ осциллограмм, который может произвести человек с достаточно высокой квалификацией. С другой стороны, если на вход правильно работающей проверяемой схемы подать некоторую эталонную циклическую стимулирующую последовательность, то осциллограмма сигнала, снимаемого с ее выхода, будет совершенно однозначно повторяться от цикла к циклу [51]. Таким образом, создание метода, позволяющего сжимать последовательность битов логических состояний на выходе схемы в ключевой код, который с высокой степенью достоверности отличается от ключевого кода любой другой последовательности, позволило единственным образом характеризовать выход проверяемой схемы.

Сжатие последовательности данных производится с жесткой синхронизацией, которая точно определяет временно́й интервал измерения – "Окно измерения" (ОИ) – и стробирует прием каждого разряда входной последовательности данных в ГПЧ сигнатурного анализатора, с помощью которого производится сжатие входной информации.

Окном измерения именуется временно́й интервал внутри циклической стимулирующей последовательности, расположенный между сигналами "Пуск" и "Стоп", которые определяют моменты начала и конца ОИ внутри стимулирующей последовательности и которые должны быть синхронны с синхронизирующим импульсом (СИ), стробирующим прием данных в ГПЧ сигнатурного анализатора. Обычно сигналы "Пуск", "Стоп", СИ поступают в СА с проверяемого устройства.

В большинстве существующих в настоящее время сигнатурных анализаторов используется ГПЧ на базе сдвигового регистра с линейными обратными связями из 16, 12, 9 и 7 разрядов, которые суммируются по модулю 2 с разрядами входной последовательности данных (рис. 3.1) [25, 51]. Этот ГПЧ был использован фирмой Hewlett Packard в первом сигнатурном анализаторе и применяется в большинстве других СА.

Из табл. 3.1, поясняющей работу схемы, видно, что первые семь разрядов входной двоичной последовательности принимаются в сдвиговый регистр без изменений, последующие же разряды, поступающие в сдвиговый регистр, являются результатом суммирования по модулю 2 разрядов входной последовательности данных и разрядов обратной связи. Число, остав-

Таблица 3.1

Выдвигаемая информация	Содержимое сдвигового регистра																Разряды входной последовательности
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
																1	1
															1	1	1
														1	1	1	1
													1	1	1	1	1
												1	1	1	1	1	1
											1	1	1	1	1	1	1
										1	1	1	1	1	1	0	0
									1	1	1	1	1	1	0	1	0
								1	1	1	1	1	1	0	1	1	0
							1	1	1	1	1	1	0	1	1	0	0
						1	1	1	1	1	1	0	1	1	0	0	0
					1	1	1	1	1	1	0	1	1	0	0	1	1
				1	1	1	1	1	1	0	1	1	0	0	1	0	1
			1	1	1	1	1	1	0	1	1	0	0	1	0	1	1
		1	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1
	1	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	1
1	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	1
1 1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1
1 1 1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1
1 1 1 1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1	1

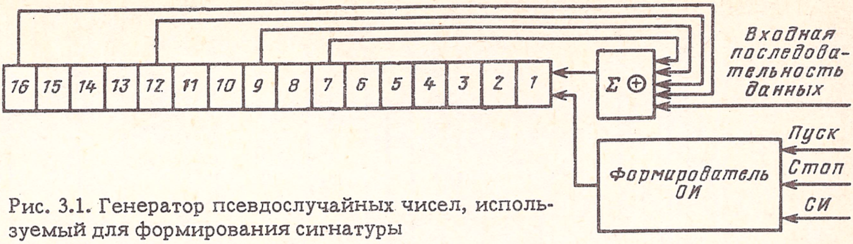


Рис. 3.1. Генератор псевдослучайных чисел, используемый для формирования сигнатуры

шееся в сдвиговом регистре после приема последнего разряда входной последовательности внутри ОИ, является сигнатурой.

Универсальные сигнатурные анализаторы HP5004A [25], микросистемный анализатор [29], 7201 Locator [35] имеют по три управляющих сигнала: "Пуск", "Стоп", "Синхронизация" (СИ). Однако для диагностирования МПК, в состав которых входят самые разнообразные устройства, недостаточно возможностей, обеспечиваемых этими сигналами по формированию ОИ и стробированию поступающих данных. Для увеличения гибкости при формировании ОИ и синхронизации в СА обеспечены новые функции, а именно:

Ожидание	– реализуется дополнительным синхронизирующим сигналом, не участвующим в формировании ОИ, а блокирующим прием данных в ГПЧ СА;
Сдвиг включения ОИ относительно сигнала "Пуск"	– сдвиг начала ОИ относительно сигнала "Пуск" на заданное оператором число СИ;
Выключение ОИ относительно сигнала "Пуск"	– формирование конца ОИ относительно сигнала "Пуск" через заданное количество СИ;
Включение и выключение ОИ по кодовым комбинациям	– формирование ОИ по кодовым комбинациям, задаваемым дополнительными управляющими сигналами, принимаемыми с проверяемого устройства.

Блок-схема такого устройства сигнатурного анализа изображена на рис. 3.2 [52]. Сигнатурный анализатор обладает следующими достоинствами: стабильным формированием ОИ для большинства цифровых устройств как синхронных, так и асинхронных; временным разделением принимаемых сигналов

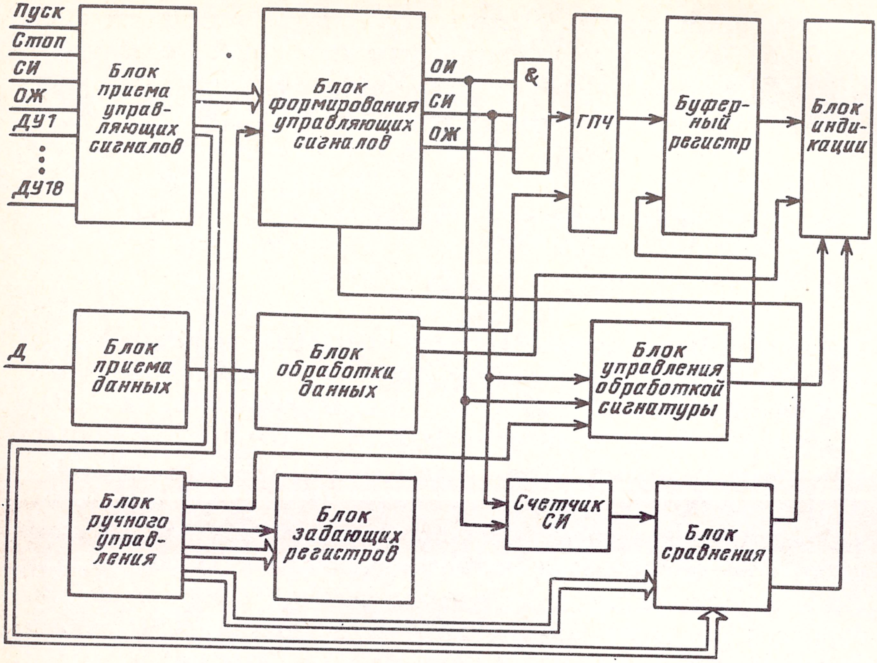


Рис. 3.2. Блок-схема СА, адаптированного для работы с МПК

данных внутри одного ОИ; приемом только определенных, в том числе и асинхронных, сигналов, принимаемых, например, с контроллеров последовательных интерфейсов. Это позволяет организовать проверки на качественно новом уровне.

В СА производится также проверка уровня напряжения на выводе проверяемой ИС, и если во время рабочего перепада СИ напряжение на выходе ИС находится в пределах 0,8–2 В, то включается светодиодный индикатор, сигнализирующий о неопределенности логического состояния на выводе проверяемой ИС.

Сигнатурный анализатор должен однозначно трактовать неопределенное логическое состояние на выводе проверяемой ИС при приеме данных в ГПЧ. При приеме такого сигнала в ГПЧ передается бит данных с логическим состоянием, соответствующим последнему принятому стандартному уровню ТТЛ-сигнала.

Сигнатурные анализаторы [25, 29, 35, 33, 34] являются универсальными приборами, которые могут быть использованы при

поиске неисправности в любых микропроцессорных устройствах. Однако универсальность рассматриваемых сигнатурных анализаторов затрудняет их использование при поиске неисправности в МПК, поскольку:

усложнена стимуляция проверяемых устройств, необходима замена БИС ПЗУ, содержащих рабочие программы на БИС ПЗУ с ТПСА, а также ввода ТПСА в ОЗУ; более поздние модификации сигнатурных анализаторов: HP 5005B [34] и 7201 Locator [35] позволяют производить стимуляцию проверяемых устройств через разъем микропроцессора, однако этот способ стимуляции можно применить в основном при поиске неисправности в центральных процессорных устройствах;

затруднено формирование ОИ и стробирование принимаемых данных из-за ограниченного количества управляющих и синхронизирующих сигналов;

использование собственного блока питания требует наличия в шкафу МПК или поблизости от него источника питающего напряжения требуемого номинала;

МПК чаще всего устанавливаются в шкафах, имеющих большую высоту, поэтому использование универсального прибора затруднено из-за отсутствия подставки, на которую он может быть установлен.

Сигнатурный анализатор, применяемый в составе МПК, должен быть как конструктивно, так и структурно приспособлен для работы в МПК.

Для решения задачи размещения сигнатурного анализатора при работе с МПК, установленным в шкафу, а также электропитания анализатора его необходимо выполнить конструктивно таким образом, чтобы он мог устанавливаться в разъемы ММ или дополнительные разъемы удлинителя.

На рис. 3.3 представлен вариант конструктивного решения СА в виде двух печатных плат, одна из которых представляет собой СА и вставляется в разъемы блока МПК, а вторая крепится перпендикулярно к первой и содержит органы управления, цифровые индикаторы, через нее передаются в СА управляющие сигналы и данные. Проверяемая плата устанавливается на удлинитель.

На рис. 3.4 представлен второй вариант конструктивного решения этой адачи. Сигнатурный анализатор выполнен в этом случае в виде стандартной печатной платы, имеющей двусторонний монтаж элементов. Основным устройством, позволяю-

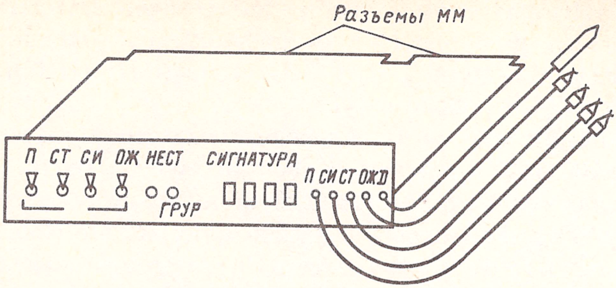


Рис. 3.3. Вариант конструктивного решения СА

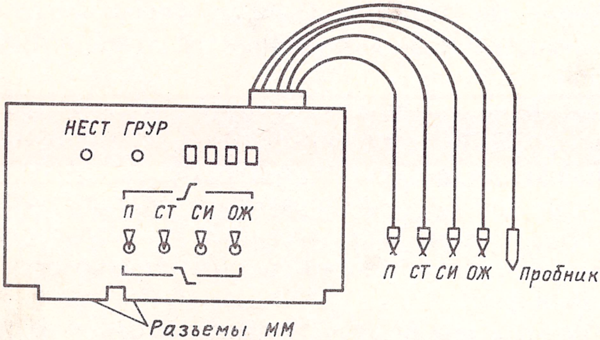


Рис. 3.4. Второй вариант конструктивного решения СА

щим решить поставленную задачу, является удлинитель, имеющий два ряда разъемов, в которые могут вставляться платы, – вертикальный и горизонтальный. В горизонтальный ряд разъемов вставляется проверяемое устройство, а в вертикальный – СА. Сам же удлинитель вставляется в разъемы ММ МПК.

Использование этих конструктивных вариантов позволяет:

использовать в качестве блока питания блок питания МПК;

управлять включением и выключением различных частей проверяемых устройств при наличии свободных контактов на разъеме блока ММ с помощью управляющих сигналов, вырабатываемых в СА;

обеспечить удобство работы с СА независимо от высоты расположения проверяемого устройства в шкафу МПК – при этом отпадает необходимость в специальном столе или подставке для диагностического прибора;

удобно расположить индикаторы сигнатуры и органы управления СА прямо перед оператором, осуществляющим поиск неисправности;

осуществить благодаря наличию доступа к межмашинной магистрали (ММ) стимулирование проверяемых устройств с помощью тестовых последовательностей, которые хранятся в собственной памяти анализатора.

Предложенные для реализации данной схемы конструктивные решения позволяют автоматизировать процесс поиска неисправности с применением сигнатурных анализаторов. Наличие у СА непосредственной связи с ММ делает его одним из устройств МПК. Поэтому преобразование СА в микропроцессорное устройство, обладающее собственной оперативной и постоянной памятью, органами управления, средствами отображения и доступом к ЦПУ через микропроцессорные разъемы, совершенно изменяет процедуру локализации неисправности. Разработанные в настоящее время БИС оперативной и постоянной памяти большой емкости позволяют при небольших затратах организовать рациональную проверку. Модификация СА позволит сократить объем документации (становятся ненужными таблицы эталонных сигнатур), упростить процесс поиска неисправности и сократить время поиска, что достигается за счет автоматизации анализа результата проверки, управления действиями оператора в соответствии с алгоритмом диагностирования и предварительным определением неисправного устройства с помощью функциональных тестов.

Блок-схема модифицированного СА представлена на рис. 3.5. В качестве схемы, обеспечивающей связь с проверяемыми ЦПУ, в таком СА может быть применена схема [53], не требующая, в отличие от внутрисхемных эмуляторов, введения дополнительных тактов ожидания для установления связи с проверяемым устройством.

Параметры сигнатурных анализаторов. СА представляет собой компактный переносной измерительный прибор, который подсоединяется к проверяемому устройству только на время его проверки.

С точки зрения локализации отказов в МПК необходимо отметить некоторые свойства СА:

1. Обеспечение гибкости при выборе управляющих сигналов "Пуск", "Стоп", СИ путем выбора рабочего перепада

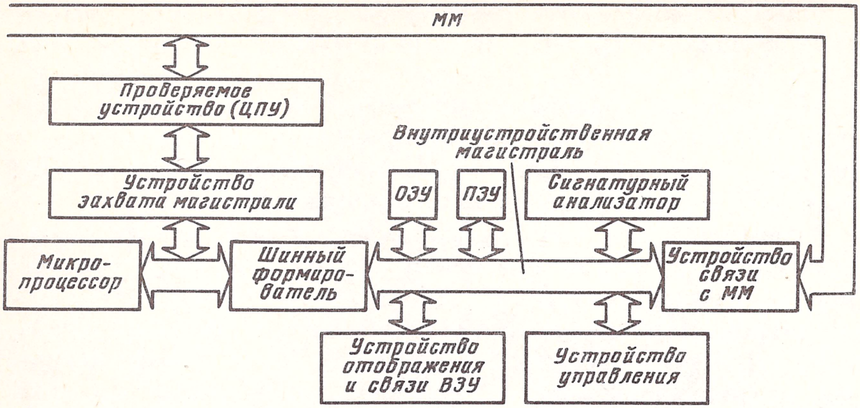


Рис. 3.5. Блок-схема автоматизированного СА

(фронта или среза) сигнала, используемого в качестве СИ, и рабочего уровня сигналов "Пуск" и "Стоп".

2. Возможность обнаружения сбоев путем сравнения сигнатур, полученных в результате двух последовательных во времени ОИ. В случае их несравнения включается сигнальный светодиодный индикатор, свидетельствующий о нестабильности работы проверяемой схемы.

3. Возможность определения соответствия уровня напряжения на выводе проверяемой ИС, стандартным уровням Лог. 0 и Лог. 1 для ТТЛ-схем. В СА производится проверка уровня напряжения на выводе проверяемой ИС, и если во время рабочего перепада СИ напряжение на выводе ИС находится в пределах 0,8–2 В, то включается светодиодный индикатор, сигнализирующий о неопределенности логического состояния на выводе проверяемой ИС.

4. Однозначность трактовки неопределенного логического состояния на выводе проверяемой ИС при приеме данных в ГПЧ.

Одним их важных параметров сигнатурных анализаторов является сигнатура. Необходимость создания методов расчета сигнатур связана, во-первых, с оценкой достоверности сигнатурных анализаторов, а во-вторых, с необходимостью автоматизации разработки систем диагностирования МПК с использованием сигнатурного анализа.

Получение сигнатуры в сигнатурном анализаторе производится с помощью генератора псевдослучайных чисел, выполненного на базе сдвигового регистра с линейными обратными связями, которые суммируются по модулю 2 с разрядами входной последовательности данных. Поведение такой схемы может быть описано с помощью математического аппарата теории полей.

При описании работы генератора псевдослучайных чисел предполагается, что на его вход поступает входная двоичная последовательность данных, которая может быть записана в виде многочлена $a(X)$ над полем характеристики 2:

$$a(X) = a_g X^g + a_{g-1} X^{g-1} + \ldots + a_1 X + a_0 = \sum_{i=0}^{g} a_i X^i, \tag{3.1}$$

где a_i – коэффициенты, принимающие значения 0 или 1. Сдвиговый регистр с обратными связями описывается полиномом обратных связей [8]:

$$P_n(X) = b_n X^n + b_{n-1} X^{n-1} + \ldots + b_1 X + b_0 = \sum_{j=0}^{n} b_j X^j, \tag{3.2}$$

где n – длина сдвигового регистра; b_j – коэффициент наличия обратной связи, принимающий значения: 0 – если в данном разряде обратная связь отсутствует, 1 – если из данного разряда взята обратная связь.

Многочлен $a(X)$ поступает на вход сумматора по модулю 2, начиная с коэффициентов высших порядков. При этом генератор псевдослучайных чисел производит преобразование многочлена (3.1), которое определяется как деление по модулю 2 многочлена $a(X)$ на инверсию полинома обратных связей $P'_n(X)$ [32, 55, 65]:

$$P'_n(X) = X^{-n} P_n(X) = b_n X^0 + b_{n-1} X +$$

$$+ \ldots + b X^{n-1} + b_0 X^n = \sum_{\substack{j=n \\ k=0}}^{\substack{j=0 \\ k=n}} b_j X^k. \tag{3.3}$$

Многочлен (3.1) и полиномы (3.2) и (3.3) должны быть нормированными, поскольку, если коэффициент при высшем порядке многочлена или полинома обратных связей равен 0, он не принимает участия в преобразовании.

Деление многочлена (3.1) на полином (3.3) можно записать в виде формулы деления многочленов:

$$\frac{a(X)}{P'_n(X)} = q_n(X) + \frac{r_n(X)}{P'_n(X)}, \tag{3.4}$$

где $q_n(X)^n$ – частное от деления; $r_n(X)$ – остаток от деления.

При этом

$$q_n(X) = c_{g-n}X^{g-n} + c_{g-n-1}X^{g-n-1} + \ldots + c_1 X + c_0; \tag{3.5}$$

$$r_n(X) = d_{n-1}X^{n-1} + d_{n-2}X^{n-2} + \ldots + d_1 X + d_0, \tag{3.6}$$

где c и d – коэффициенты, принимающие значения 0 или 1.

Из (3.4) и (3.5) следует, что $q_n(X)$ соответствует разрядам выходной двоичной последовательности, выдвигаемым из сдвигового регистра, а $r_n(X)$ представляет собой двоичное число, оставшееся в сдвиговом регистре. Однако остаток $r_n(X)$, полученный, например, в результате деления углом $a(X)$ на $P'_n(X)$, не будет равен реальному числу (сигнатуре), оставшемуся в сдвиговом регистре после прохождения через него входной двоичной последовательности, соответствующей (3.1).

Для получения сигнатуры остаток $r_n(X)$ необходимо преобразовать некоторым образом. Для этого по методу, описанному в [65], необходимо произвести умножение остатка $r_n(X)$ на квадратную матрицу. Причем эта матрица индивидуальна для каждого сдвигового регистра.

Другим важным параметром СА является его достоверность, которая определяется достоверностью используемого ГПЧ.

Под достоверностью СА понимается вероятность, с которой две различные входные последовательности, поступающие на вход СА, дадут различные сигнатуры. Анализ достоверности СА производится в ряде отечественных и зарубежных публикаций [1, 51, 31, 61, 63].

В рассмотренных публикациях показывается высокая достоверность СА, однако не производится сравнение СА с генераторами псевдослучайных чисел, которые выполнены на базе различных характеристических полиномов, т.е. отличающихся как по длине сдвиговых регистров, так и по обратным связям, не вскрывается взаимосвязь между достоверностью, количеством и расположением обратных связей в сдвиговом регистре. Такой анализ необходим для разработки рекомендаций по оптимальному выбору ГПЧ для сигнатурного анализатора, предназначенного для поиска неисправностей в конкретном МПК. Эта задача является актуальной, поскольку наряду с ГПЧ, разработанным фирмой Hewlett Packard для своего первого сигнатурного анализатора HP5004A, который стал прототипом для большинства сигнатурных анализаторов, разрабатываются СА с другими ГПЧ, например СА 1310 А фирмы М.В. Electronik [33].

3.2. РАЗРАБОТКА МЕТОДА РАСЧЕТА СИГНАТУР И АЛГОРИТМОВ ИХ ВЫЧИСЛЕНИЯ С ПОМОЩЬЮ ЭВМ

Пусть на вход генератора псевдослучайных чисел (рис. 3.6) поступает входная двоичная последовательность, которая может быть записана как входной многочлен $a(X)$ вида (3.1). Генераторы псевдослучайных чисел, подобные изображенному на рис. 3.6, осуществляют преобразование входной двоичной последовательности, соответствующее делению входного многочлена $a(X)$ на инверсию характеристического полинома, описывающего данный ГПЧ, по формуле (3.3). Характеристический полином ГПЧ должен быть примитивным [54, 31].

Предположим, что сдвиговый регистр (рис. 3.6) имеет длину n разрядов и обратные связи взяты из $n, m_1, m_2, \ldots, m_s$ разрядов, причем $n > m_1 > m_2 > \ldots > m_s$. Рассмотрим n-раз-

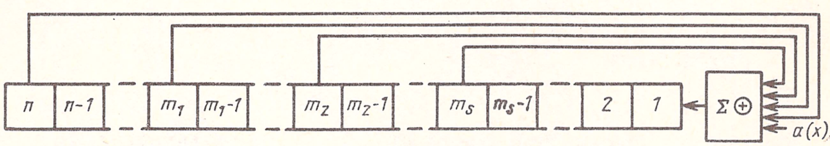


Рис. 3.6. Схема ГПЧ на базе сдвигового регистра длиной n разрядов

Таблица 3.2

Длина сдвигового регистра	Разряды, из которых взяты обратные связи	Обозначение характеристического полинома	Обозначение инверсии характеристического полинома
n	$n, m_1, m_2, \ldots, m_s$	$P_n(X)$	$P'_n(X)$
m_1	$m_1, m_2, \ldots, m_s$	$P_{m_1}(X)$	$P'_{m_1}(X)$
m_2	$m_2, \ldots, m_s$	$P_{m_2}(X)$	$P'_{m_2}(X)$
:	:	:	
m_s	m_s	$P_{m_s}(X)$	$P'_{m_s}(X)$

рядный сдвиговый регистр в виде набора сдвиговых регистров с обратными связями (табл. 3.2).

Характеристические полиномы соответствующих сдвиговых регистров и их инверсии будут равны:

$$P_n(X) = X^n + X^{m_1} + X^{m_2} + \ldots + X^{m_s} + 1;$$

$$P_{m_1}(X) = X^{m_1} + X^{m_2} + \ldots + X^{m_s} + 1;$$

$$P_{m_2}(X) = X^{m_2} + \ldots + X^{m_s} + 1;$$

$$P_{m_s}(X) = X^{m_s} + 1;$$

$$P'_n(X) = X^n + X^{n-m_s} + \ldots + X^{n-m_2} + X^{n-m_1} + 1;$$

$$P'_{m_1}(X) = X^{m_1} + X^{m_1-m_s} + \ldots + X^{m_1-m_2} + 1;$$

$$P'_{m_2}(X) = X^{m_2} + X^{m_2-m_s} + \ldots + 1;$$

$$P'_{m_s}(X) = X^{m_s} + 1.$$

Обозначим сигнатуру через $\sigma(X)$, а частные и остатки, получающиеся при делении входного многочлена по (3.4)

на инверсии характеристических полиномов соответствующих сдвиговых регистров, через:

$$q_n(X),\ q_{m_1}(X),\ q_{m_2}(X),\ q_{m_s}(X),$$

$$r_n(X),\ r_{m_1}(X),\ r_{m_2}(X),\ r_{m_s}(X).$$

Представим $a(X)$ в виде суммы двух многочленов $a_1(X)$ и $a_2(X)$, тогда в связи с линейностью полиномов согласно [54] можно утверждать, что

$$q(a(X)) = q(a_1(X) + a_2(X)) = q_1(a_1(X)) + q_2(a_2(X));$$

$$\sigma(a(X)) = \sigma(a_1(X) + a_2(X)) = \sigma_1(a_1(X)) + \sigma_2(a_2(X)). \quad (3.7)$$

Как было показано выше, остаток $r(X)$, полученный при делении $a(X)$ на $P(X)$ по (3.4), отличен от реальной сигнатуры. Поэтому важно установить взаимосвязь между сигнатурой входного многочлена $a(X)$ и сигнатурой остатка $r(X)$.

Теорема 1. Для всякого $a(X)$ существует единственный полином $r_n(X)$ степени не более $n-1$, такой, что сигнатура

$$\sigma(a(X)) = \sigma(r_n(X)). \quad (3.8)$$

Доказательство. Представим $a(X)$ (3.1) в виде суммы двух многочленов $q_n(X)\,P_n'(X)$ и $r_n(X)$. Тогда из (3.4) и (3.7) следует

$$a(X) - r_n(X) = q_n(X)\,P_n'(X). \quad (3.9)$$

Известно [54, 62], что если $q_n(X)$ – выходной многочлен, то

$$\sigma_n(q_n(X)\,P_n'(X)) = 0. \quad (3.10)$$

Из (3.10) и (3.7) следует

$$0 = \sigma_n(q_n(X)\,P_n'(X)) = \sigma(a(X) - r_n(X)) = \sigma(a(X)) - \sigma(r_n(X)),$$

т.е

$$\sigma(a(X)) = \sigma(r_n(X)), \quad (3.11)$$

что и требовалось доказать.

Сформулируем теорему о вычислении сигнатуры с использованием метода вложенных полиномов [11].

Теорема 2. Пусть $a(X)$ является входным многочленом. Тогда сигнатура, полученная при делении $a(X)$ на полином $P'_n(X)$, будет равна

$$\sigma_n(X) = q_{m_1}(X) X^{m_1+1} + q_{m_2}(X) X^{m_2+1} + \ldots$$

$$\ldots + q_{m_s}(X) X^{m_s+1} + r_{m_s}(X) X. \tag{3.12}$$

Доказательство. Запишем входной многочлен $a(X)$ в виде

$$a(X) = q_n(X) P'_n(X) + r_n(X).$$

По теореме 1

$$\sigma(a(X)) = \sigma(r_n(X)),$$

т.е. сигнатуры многочленов $a(X)$ и $r_n(X)$ равны. Поскольку степень $r_n(X)$ меньше степени $P'_n(X)$, то деление многочлена $r_n(X)$ производится на вложенный по отношению к $P'_n(X)$ полином $P'_{m_1}(X)$, степень которого меньше степени $r_n(X)$.

Согласно [55] $q_n(X)$ является выходным многочленом.

Члены сигнатуры со степенями n, $n-1$, ..., m_1+1 определяются частным от деления $r_n(X)$ на $P'_{m_1}(X)$.

Сигнатура может быть записана в виде

$$\sigma_n(X) = q_{m_1}(X) X^{m_1+1} + t_{m_2}(X), \tag{3.13}$$

где $t_{m_2}(X)$ – неизвестная часть сигнатуры, а q_{m_1} получим из формулы

$$r_n(X) = q_{m_1}(X) P'(X) + r_{m_1}(X). \tag{3.14}$$

Известно, что $r_m(X)$ имеет степень, меньшую m_1 и что степень $t_{m_2}(X)$ из (3.13) также меньше m_1, следовательно, $r_{m_1}(X)$ можно рассматривать в качестве входного многочлена для

полинома $P'_{m_2}(X)$. Разделим многочлен $r_{m_1}(X)$ на вложенный по отношению к $P_{m_1}(X)$ полином $P'_{m_2}(X)$. В результате деления получим

$$r_{m_2}(X) = r_{m_1}(X) - q_{m_2}(X) P'_{m_2}(X). \tag{3.15}$$

Частное от деления $q_{m_2}(X)$ (3.15) определит степени m_1, $m_1 - 1, \ldots, m_2 + 1$ слагаемого $t_{m_2}(X)$ в (3.13). Отсюда

$$t_{m_2}(X) = q_{m_2}(X) X^{m_2+1} + t_{m_3}(X),$$

где $t_{m_3}(X)$ – неизвестная часть сигнатуры.

Процесс можно продолжить. В результате взаимных подстановок получим

$$\sigma_n(X) = q_{m_1}(X) X^{m_1+1} + q_{m_2}(X) X^{m_2+1} + \ldots$$
$$\ldots + q_{m_s}(X) X^{m_s+1} + r_{m_s}(X) X,$$

что и требовалось доказать. Теорема 2 может быть названа методом вложенных полиномов.

Выражение (3.12) является математической моделью работы ГПЧ (см. рис. 3.6) и может быть использовано для вычисления сигнатур в любых ГПЧ рассмотренного типа. В связи с тем что нумерация регистров обычно начинается с нуля, то (3.12) для данной нумерации можно переписать следующим образом:

$$\sigma^1_n(X) = q_{m_1}(X) X^{m_1} + q_{m_2}(X) X^{m_2} + \ldots$$
$$\ldots + q_{m_s}(X) X^{m_s} + r_{m_s}(X). \tag{3.16}$$

Для получения формулы расчета сигнатуры в конкретном ГПЧ в выражении (3.16) вместо X^i, где $i = m_1, m_2, \ldots, m_s$ необходимо подставить номера разрядов сдвигового регистра ГПЧ, из которых взяты обратные связи.

Например, в соответствии с (3.16) для ГПЧ (см. рис. 3.1) формула расчета сигнатуры имеет вид

$$\sigma_{16}^{HP}(X) = q_{12}(X)X^{12} + q_9(X)X^{9} + q_7(X)X^{7} + r_7(X). \tag{3.17}$$

В качестве примера применения метода вложенных полиномов определим сигнатуру для входной последовательности, указанной в [25].

Приведем входную двоичную последовательность 1111110000011 1111111 к виду многочлена в соответствии с весами двоичных разрядов:

$$a_{19}(X) = X^{19} + X^{18} + X^{17} + X^{16} + X^{15} + X^{14} + 0 + 0 + 0 + 0 +$$

$$+ 0 + X^{8} + X^{7} + X^{6} + X^{5} + X^{4} + X^{3} + X^{2} + X + 1. \tag{3.18}$$

Разделим многочлен (3.18) на инверсию полинома обратных связей сдвигового регистра с обратными связями (см. рис. 3.1), которая, как известно из [54], равна

$$P'_{16}(X) = X^{16} + X^{9} + X^{7} + X^{4} + 1. \tag{3.19}$$

В результате деления (3.18) на (3.19) получим остаток

$$r_{16}(X) = X^{15} + X^{14} + 0 + X^{12} + X^{11} +$$

$$+ 0 + 0 + 0 + X^{7} + 0 + 0 + 0 + 0 + 0 + 0 + 0 \tag{3.20}$$

и частное

$$q_{16}(X) = X^{3} + X^{2} + X + 1. \tag{3.21}$$

Затем полученный остаток разделим на инверсию полинома обратных связей 12-разрядного регистра:

$$P'_{12}(X) = X^{12} + X^{5} + X^{3} + 1. \tag{3.22}$$

В результате деления (3.20) на (3.22) получим частное

$$q_{12}(X) = X^{3} + X^{2} + 0 + 1 \tag{3.23}$$

и остаток

$$r_{12}(X) = X^{11} + 0 + 0 + X^{8} + 0 + X^{6} + 0 + 0 + 0 + X^{2} + 0 + 1, \qquad (3.24)$$

который разделим на

$$P'_{9}(X) = X^{9} + X^{2} + 1. \qquad (3.25)$$

В результате деления получим частное

$$q_{9}(X) = X^{2} + 0 + 0 \qquad (3.26)$$

и остаток

$$r_{9}(X) = X^{8} + 0 + X^{6} + 0 + X^{4} + 0 + 0 + 0 + 1. \qquad (3.27)$$

И на последнем этапе разделим остаток (3.27) на инверсию полинома обратных связей 7-разрядного регистра

$$P'_{7}(X) = X^{7} + 1. \qquad (3.28)$$

Заметим, что инверсия полинома обратных связей сдвигового регистра с одной линейной обратной связью совпадает с его полиномом обратных связей.

В результате деления остатка (3.27) на полином (3.28) получим частное

$$q_{7}(X) = X + 0 \qquad (3.29)$$

и остаток

$$r_{7}(X) = X^{6} + 0 + X^{4} + 0 + 0 + X + 1. \qquad (3.30)$$

Подставим полученные значения (3.23), (3.26), (3.29) и (3.30) в (3.17):

$$\sigma^{HP}_{16}(X) = (X^{3} + X^{2} + 0 + 1)X^{12} + (X^{12} + 0 + 0)X^{9} +$$

$$(X + 0)X^{7} + X^{6} + 0 + X^{4} + 0 + 0 + X + 1 = X^{15} + X^{14} +$$

$$0 + X^{12} + X^{11} + 0 + 0 + X^{8} + 0 + X^{6} + 0 + X^{4} + 0 + 0 + X + 1. \qquad (3.31)$$

Преобразуем полученный многочлен (3.31) в двоичную форму и заполним сдвиговый регистр:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1
D				9				5				3			

Таким образом, полученная сигнатура в шестнадцатеричном коде равна *D*953, т.е. равна сигнатуре, полученной в просмотренном примере.

Как показано в [33], одним из препятствий на пути широкого внедрения сигнатурного анализа является большой объем текстовой документации в виде словарей эталонных сигнатур, составление которых в настоящее время производится в основном вручную. Словарь эталонных сигнатур представляет собой документ, в котором в соответствии с алгоритмом проверки записаны эталонные сигнатуры для различных выводов ИС проверяемого устройства, кроме того, в нем записываются номера выводов, с которых должны подаваться в СА управляющие и синхронизирующие сигналы, и некоторая другая информация. Для автоматизации процесса составления этого документа необходима программа, позволяющая осуществить автоматический расчет эталонных сигнатур.

Эффективный алгоритм расчета сигнатур может быть создан на основе метода вложенных полиномов, блок-схема алгоритма представлена на рис. 3.7.

По данному алгоритму разработана программа [57] расчета сигнатур для ГПЧ (см. рис. 3.1), блок-схема которой изображена на рис. 3.8.

Расчет эталонной сигнатуры начинается с формирования входной последовательности, которое осуществляет модуль *1*. Этот модуль может быть как программой, моделирующей работу проверяемого устройства и осуществляющей автоматическое формирование входной последовательности, так и программой, преобразующей последовательность, заданную оператором, в двоичную форму и размещающей ее в памяти.

После окончания формирования входной последовательности управление передается модулю *2*, который осущест-

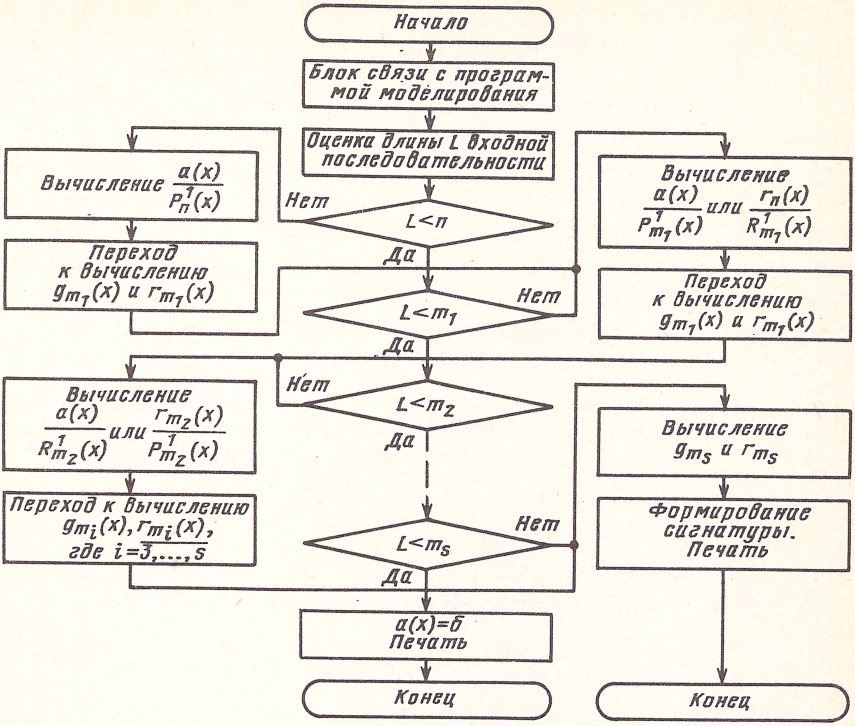


Рис. 3.7. Блок-схема алгоритма вычисления сигнатуры

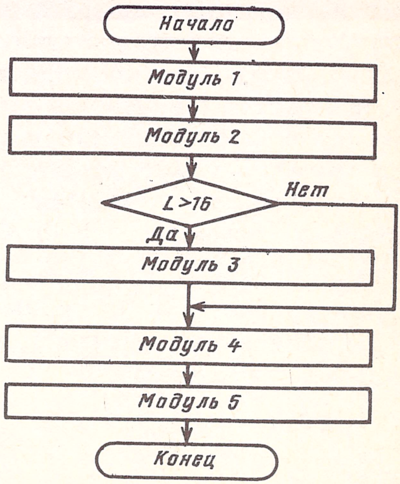


Рис. 3.8. Блок-схема программы расчета эталонных сигнатур

вляет ее анализ, т.е. определяет длину входной последовательности и загружает регистры и счетчики микропроцессора вычислительного устройства. Затем в зависимости от длины входной последовательности управление передается либо модулю *3*, либо модулю *4*.

Модуль *3* осуществляет деление входной последовательности на константу 81488_{16}, которая соответствует полиному (3.19). Деление осуществляется по алгоритму деления столбиком. Модуль *3* обрабатывает входные последовательности длиной 17 бит и более.

После окончания работы модуля *3* управление передается модулю *4*, который осуществляет деление входной последовательности на константы 8148_{16}, 8140_{16} и 81_{16}, которые соответствуют полиномам (3.22), (3.25), (3.28).

Сформированная сигнатура обрабатывается модулем *5* и выводится на печать.

3.3. ОЦЕНКА ДОСТОВЕРНОСТИ СИГНАТУРНЫХ АНАЛИЗАТОРОВ С РАЗЛИЧНЫМИ ГЕНЕРАТОРАМИ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

В проводившихся ранее исследованиях достоверности рассматривались векторы правильных входных последовательностей и векторы ошибочных входных последовательностей с последующим их суммированием. Ниже приведен другой подход, основанный на анализе всего множества векторов входных последовательностей с использованием математического аппарата линейной алгебры. В результате получены достаточно интересные результаты, которые позволяют не только установить достоверность данного ГПЧ и произвести сравнение достоверностей для различных ГПЧ, но и при необходимости вычислить по полученной сигнатуре множество векторов входной последовательности, используя взаимные соответствия между сигнатурой и векторами входных последовательностей, отображенные в табл. 3.3.

Многочлены $a_i(X) \in A$, подаваемые на вход ГПЧ, образуют кольцо [55]. Выделим в множестве A подмножество V многочленов, кратных $P'_n(X)$, каждый член которого удовлетворяет условию

$$a_j(X) = q_{nj}(X) P'_n(X) + 0.$$

Таблица 3.3

Образующий класса вычетов $r_n(X)$	Классы вычетов (входные последовательности, приведенные к виду $a(X) = P'(X)\,q(X) + r(X)$)			
	$P = n+1$	$P = n+2$	...	$P = P$
0	$P'(X)$	$P'(X)X$; $P'(X)(X+1)$	...	$P'(X)(X^p + X^{p-1} + \ldots$ $\ldots + X + 1)$
1	$P'(X)+1$	$P'(X)X+1$; $P'(X)(X+1)+1$	...	$P'(X)(X^p + X^{p-1} + \ldots$ $\ldots + X + 1) + 1$
2	$P'(X)+2$	$P''(X)X+2$; $P'(X)(X+1)+2$	...	$P'(X)(X^p + X^{p-1} + \ldots$ $\ldots + X + 1) + 2$
...	...	...	...	...
2^{n-1}	$P'(X)+$ $+2^{n-1}$	$P'(X)X+2^{n+1}$; $P'(X)(X+1)+2^{n-1}$	...	$P'(X)(X^p + X^{p-1} + \ldots$ $\ldots + X + 1) + 2^{n-1}$

Подмножество V является идеалом [63]. Образуем в множестве A классы вычетов по идеалу V; образующими которых являются остатки от деления многочленов $a_i(X) \in A$ на $P_n'(X)$. Поскольку полином $P_n'(X)$ по определению [2, 7, 8] должен быть неприводимым, то все возможные остатки $r_{ni}(X)$ будут соответствовать множеству всех двоичных чисел от 0 до 2^{n-1}. Сведем классы вычетов в табл. 3.3.

В теореме 1 установлено соответствие между $\sigma(a(X))$ и $\sigma(r_n(X))$, а по теореме 2 можно вычислить $\sigma(r_n(X))$ для любого $r_{ni}(X)$. Таким образом, табл. 3.3 является отображением множества A в множество сигнатур Σ.

В табл. 3.3 выходные многочлены приведены к виду

$$a_i(X) = q_{ni}(X) P_n^1(X) + r_{ni}(X).$$

Длина $a_i(X)$ от 0 до P разрядов ($P = 2^n$ разрядов).

Из полученной табл. 3.3 наглядно видно, что две различные входные последовательности будут неразличимы (т.е. дадут одну и ту же сигнатуру) только в том случае, если они принадлежат одному классу вычетов. Поскольку для полинома $P'_n(X)$ степени n существует n классов вычетов, то вероятность, с которой две произвольные входные последовательности будут неразличимы, равна

$$B = 2^{P-n}/2^{P} = 1/2^{n}. \tag{3.32}$$

Достоверность сигнатурного анализатора определяется как вероятность, с которой две различные входные последовательности дадут разные сигнатуры, т.е. как вероятность, с которой эти последовательности будут принадлежать различным классам вычетов. Достоверность СА равна

$$D = (1 - B)100\% = (1 - 1/2^{n})\,100\%, \tag{3.33}$$

причем формулы (3.32) и (3.33), очевидно, следуют из табл. 3.3 [60]. Более того, анализ выражения (3.33) и табл. 3.3 показывает, что достоверность сигнатурного анализатора зависит только от длины сдвигового регистра и не зависит от выбора разрядов обратной связи (при условии, что полином $P_n(X)$ неприводим). Для 16-разрядного сдвигового регистра она равна 99,998475%, а для 24-разрядного регистра 99,9999941%. Важно отметить, что для входных последовательностей длиной, равной или меньшей n, достоверность равна 100%.

В связи с тем что некоторые устройства могут иметь ошибки определенного вида (например, пропуск части информации, потеря одного слова и т.д.), необходимо иметь возможность выбрать для сигнатурного анализатора, работающего с этими устройствами, такой ГП4, для которого эти ошибки были бы неразличимы. Предлагаемая таблица позволяет произвести вычисление входного многочлена по полиному сдвигового регистра ГПЧ, частному и остатку, выполнить обратные преобразования, т.е. позволяет осуществить правильный выбор ГПЧ.

Рассмотрим пример для ГПЧ (см. рис. 3.1). Определим входные последовательности, имеющие, например, длину 18 бит, которые принадлежат классу вычетов, имеющему образующий элемент, равный 0. Для этого умножим полином

$P^1_{16}(X)$ (3.19) на все возможные значения $q(X)$ для последовательности длиной 18 бит в данном классе и получим

$$a_1(X) = P^1_{16}(X)X = X^{17} + X^{10} + X^8 + X^5 + X; \tag{3.34}$$

$$a_2(X) = P^1_{16}(X)(X+1) = X^{17} + X^{16} + X^{10} +$$

$$+ X^9 + X^8 + X^7 + X^5 + X^4 + X + 1. \tag{3.35}$$

Произведем сложение (3.34) и (3.35) по модулю 2:

$$a_1(X) \otimes a_2(X) = X^{16} + X^9 + X^7 + X^4 + 1. \tag{3.36}$$

Таким образом, входные последовательности (3.34) и (3.35) отличаются в пяти разрядах и, более того, их сумма равна инверсии полинома обратных связей.

Следовательно, для того чтобы ошибка осталась неразличимой для рассмотренного в примере ГПЧ, необходимо, чтобы она произошла в пяти разрядах 18-разрядной входной последовательности и они были бы кратны инверсии полинома обратных связей сдвигового регистра.

Для сравнения рассмотрим ГПЧ (см. рис. 3.9), выполненный на базе другого неприводимого многочлена, выбранного произвольно из [54] (табл. В2, степень 16, номер полинома 1639). Приведем полином обратных связей 271341, заданный в восьмеричном коде, к виду полинома $P_n(X)$:

$$P_{16_1}(X) = X^{16} + X^{15} + X^{14} + X^{13} + X^{10} + X^8 + X^7 + X^6 + 1. \tag{3.37}$$

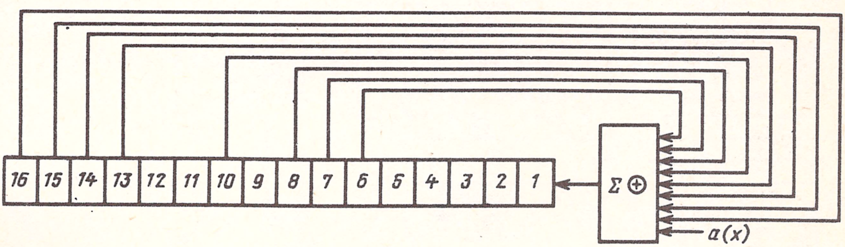


Рис. 3.9. ГПЧ на базе сдвигового регистра, имеющего полином 271341

Инверсия полинома обратных связей (3.37) будет равна

$$P^{1}_{16_1}(X) = X^{16} + X^{10} + X^{9} + X^{8} + X^{6} + X^{3} + X^{2} + X + 1. \qquad (3.38)$$

Как следует из (3.33), полученный сигнатурный анализатор будет обладать той же достоверностью, что и сигнатурный анализатор с ГПЧ на рис. 3.1, поскольку степень полинома не изменилась. Возможность распознавания двойных ошибок у этих анализаторов также одинакова, поскольку полиномы их ГПЧ принадлежат к одной и той же экспоненте $e^{16} - 1$.

Для ГПЧ на рис. 3.10, рассмотрим входные двоичные последовательности длиной 18 разрядов, принадлежащие классу вычетов, образующий элемент которого равен нулю. Для этого произведем умножение (3.38) на все возможные

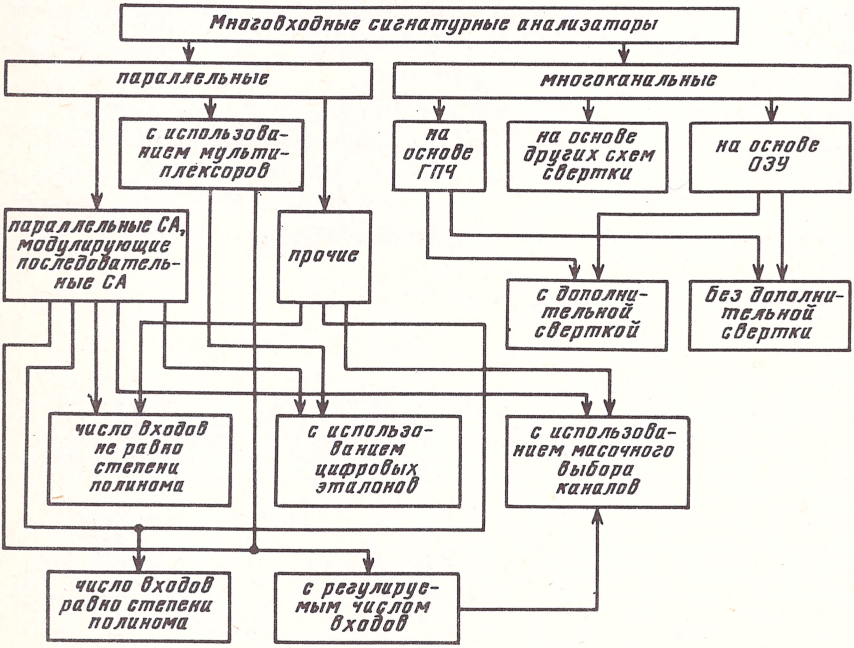


Рис. 3.10. Классификация многовходовых сигнатурных анализаторов

$q(X)$ и получим

$$a_1^1(X) = P_{16_1}^1(X)X = X^{17} + X^{11} + X^{10} + X^9 +$$

$$+ X^7 + X^4 + X^3 + X^2 + 1; \quad (3.39)$$

$$a_2^1(X) = P_{16_1}^1(X)(X+1) = X^{17} + X^{16} +$$

$$+ X^{11} + X^8 + X^7 + X^6 + X^4 + 1. \quad (3.40)$$

Сложив (3.39) и (3.40) по модулю 2, получим

$$a_1^1(X) \otimes a_2^1(X) = X^{16} + X^{10} + X^9 + X^8 + X^6 + X^3 + X^2 + X + 1. \quad (3.41)$$

Следовательно, для неразличимости ошибки (для СА) необходимо, чтобы входная двоичная последовательность, образованная в результате ее возникновения, относилась к тому же классу вычетов, что и правильная входная последовательность, другими словами, разность между правильной и ошибочной входными последовательностями должна быть кратна инверсии полинома обратных связей данного ГПЧ.

3.4. МНОГОВХОДОВЫЕ СИГНАТУРНЫЕ АНАЛИЗАТОРЫ И ИХ ПРИМЕНЕНИЕ

Современные цифровые узлы характеризуются большой сложностью, имеют большое число входов и выходов (от нескольких десятков до нескольких сотен). В связи с этим диагностические таблицы для поиска неисправностей в подобных узлах достаточно объемны и их использование во многих случаях при поиске неисправностей приводит к неоправданно большим затратам времени обслуживающего персонала.

Одним из методов ускорения поиска неисправностей в сложных многовыходных цифровых узлах является использование многовходовых сигнатурных анализаторов.

На рис. 3.10 приведена классификация многовходовых сигнатурных анализаторов. Рассмотрим их характерные

структурные особенности, конструктивные решения и преимущественные области применения.

В общем, многовходовые сигнатурные анализаторы (СА) можно подразделить на параллельные и многоканальные. Многоканальные сигнатурные анализаторы отличаются тем, что имеют множество (в частном случае равное числу контролируемого цифрового узла) идентичных каналов, каждый их которых представляет собой одновходовый СА, построенный по любой из известных схем (см. выше), т.е. синхронный или асинхронный, с контролем Z-состояния или без него, на основе тех или иных образующих полиномов и т.д.

По признаку построения отдельного канала многоканальные СА можно подразделить на СА на основе ГПЧ, СА на основе ОЗУ, СА на основе других систем свертки.

СА на основе ГПЧ представляет собой совокупность n ГПЧ традиционной структуры, каждый из которых своим информационным входом подключается к соответствующему выходу контролируемого цифрового узла. Как правило, подобные СА имеют общие цепи синхронизации и "окна измерения". В процессе контроля с помощью подобных СА осуществляется одновременное формирование n сигнатур соответствующей разрядности, которые впоследствии индивидуально сравниваются с соответствующими эталонными структурами.

Большой объем выходной информации является основным недостатком этих СА. Для преодоления указанного недостатка в ряде схемных решений используется дополнительная свертка n полученных сигнатур в одну результирующую сигнатуру. Так, на рис. 3.11 приведена структура многоканального СА, имеющего $n + 1$ формирователь сигнатур (ФС), где $(n + 1)$-й ФС используется для вспомогательной свертки полученных сигнатур в одну, которая впоследствии и сравнивается с эталонным значением.

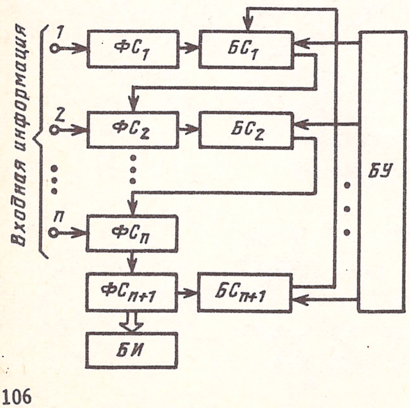


Рис. 3.11. Многоканальный СА с дополнительной сверткой информации

К общим недостаткам описанных многоканальных СА следует отнести большие аппаратные затраты, что становится особенно заметным при нескольких десятках информационных входов СА, а для нескольких сотен информационных входов подобные СА уже абсолютно неприменимы.

Одним из путей сокращения аппаратных затрат при построении многоканальных СА является использование ОЗУ в качестве основы для последовательного ГПЧ в каждом из каналов. В этом случае при использовании ОЗУ с организацией $m^x + n$ (где n – число информационных входов, а m – разрядность сигнатуры) удается сократить аппаратные затраты приблизительно в n раз в пересчете на корпуса используемых микросхем.

Многоканальные СА не имеют собственных существенных отличительных особенностей по сравнению с последовательными СА, на основе которых они построены, и их свойства являются суммой свойств составляющих их каналов. Это же справедливо и для многоканальных СА, построенных с использованием произвольных схем свертки в каждом из каналов (подсчет числа логических состояний, подсчет числа переходов, стохастические преобразования, числовые сигнатуры на основе свойств полноты логических функций и т.п.).

Другую большую группу многовходовых СА составляют так называемые параллельные СА. Для них характерно единство и непериодичность внутренней структуры, невозможность ее разбиения на идентичные составляющие.

Простейшим параллельным СА (ПСА) является ПСА на основе одновходового последовательного ГПЧ с использованием мультиплексора на входе [64]. В этом случае входные информационные последовательности от каждого из выходов контролируемого цифрового узла последовательно "выстраиваются" в единый информационный вектор, который и поступает на свертку. Так, если на n выходах контролируемого узла присутствуют последовательности длиной k:

$$\overline{A_1} = \overline{a_{11}\, a_{12}\, \ldots\, a_{1k}};$$

$$\overline{A_2} = \overline{a_{21}\, a_{22}\, \ldots\, a_{2k}};$$

.

$$\overline{A_n} = \overline{a_{n1}\, a_{n2}\, \ldots\, a_{nk}};\ a_{ij} \in \{0, 1\};\ 1 \leqslant i \leqslant n;\ 1 \leqslant j \leqslant k,$$

то на входе ПСА будет присутствовать информационная последовательность:

$$\overline{A} = \overline{\overline{A}_1\, \overline{A}_2\, \overline{A}_n} =$$

$$= \overline{a_{11}\, a_{12}\, \ldots\, a_{1k}\, a_{21}\, a_{22}\, \ldots\, a_{2k}\, \ldots\, a_{n1}\, a_{n2}\, a_{nk}}.$$

Недостатком подобного ПСА является большое время контроля, связанное с тем, что выходы контролируемого узла подключаются к ПСА по очереди и в связи с этим время контроля увеличивается в n раз, где n – число контролируемых выходов.

Большой интерес представляют собой ПСА, которые в своей структуре реализуют параллельную модель последовательного СА за n тактов работы. Структура такого СА строится исходя из следующих соображений.

Пусть нам дан последовательный СА разрядностью m. Состояние каждого из его разрядов в произвольный момент времени t_i обозначим через S_{ij}, где $1 \leqslant j \leqslant m$. Пусть образующий полином ГПЧ имеет вид $P_m(X) = \sum\limits_{k=0}^{m} a_k x^m,\ a_k \in \{0, 1\}$.

Тогда при поступлении на его вход первого бита входной информационной последовательности b_1 в первый разряд ПСА запишется информация

$$S_{11} = b_1 \oplus \sum_{k=0}^{m} a_k,$$

в остальные разряды будет записываться информация из предыдущих разрядов:

$$S_{1p} = S_{1(p-1)};\ \ 1 \leqslant p \leqslant (m-1).$$

То же будет происходить и во все последующие моменты времени. Через m тактов вся старая информация в регистре

будет заменена на новую – результат свертки входной информационной последовательности. Из алгоритма работы устройства видно, что с учетом обозначений, введенных выше, данный тип ПСА формирует на своих входах эквивалентный входной вектор:

$$\overline{A_n} = \overline{a_{11}\, a_{21}\, \ldots\, a_{n1}\, a_{12}\, a_{22}\, \ldots\, a_{n2}\, \ldots\, a_{1k}\, a_{2k}\, \ldots\, a_{nk}}\,.$$

В [65] описана система уравнений и приведена функциональная схема ПСА, реализованного на образующем полиноме $P(X) = X^{16} + X^{12} + X^{9} + X^{7} + 1$ для количества входов $K = 16$, т.е. реализована параллельная модель последовательного СА за 16 тактов работы. Однако вовсе не обязательно равенство степени образующего полинома m, количества информационных входов K и номера такта B, на котором осуществляется моделирование. Так, в [65] приведен алгоритм построения ПСА для заданного образующего полинома и произвольного количества входов. При этом $K = 16$. В то же время для любого из этих ПСА при числе выходов контролируемого элемента, меньшем числа информационных входов ПСА, возможен контроль с подачей на незадействованные входы ПСА определенной детерминированной последовательности (чаще всего нулевой). Это эквивалентно переводу ПСА на определенном такте работы в режим ГПЧ.

Принципы синтеза ПСА, представляющие собой модели последовательных СА, поясним на конкретном примере.

Пусть задан образующий полином $P(X) = X^{16} + X^{12} + X^{9} + X^{7} + 1$.

Основным узлом формирователя сигнатур является регистр разрядности, равной разрядности образующего полинома, для хранения промежуточных результатов свертки.

Опишем состояния каждого из разрядов 16-разрядного регистра при последовательной подаче на вход формирователя сигнатур бит $D_1 - D_{16}$ входной информации.

Исходное состояние регистра в момент времени t_0 :

$R_{10} = C_1$;

$R_{20} = C_2$;

$R_{30} = C_3$;

.

$R_{160} = C_{16}$.

В момент времени t_1 входной бит D_1 суммируется по модулю 2 с битами, хранящимися в 16, 12, 9 и 7-м разрядах регистра, результат записывается в первый разряд регистра, в остальные разряды записывается информация из предыдущих разрядов (сдвиг вправо). Система уравнений, описывающая состояние регистра в момент времени t_1, имеет вид

$R_{11} = D_1 \oplus C_{16} \oplus C_{12} \oplus C_9 \oplus C_7$;

$R_{21} = C_1$;

$R_{31} = C_2$;

.

$R_{161} = C_{15}$.

Здесь $\oplus$ – знак сложения по модулю 2.

Данная система описывает одновходовой СА на основе параллельного регистра и пятивходового сумматора по модулю 2. Его функциональная схема приведена на рис. 3.12.

Аналогичным образом получим систему уравнений для момента времени t_2:

$R_{12} = D_2 \oplus C_{15} \oplus C_{11} \oplus C_8 \oplus C_6$;

$R_{22} = D_1 \oplus C_{16} \oplus C_{12} \oplus C_9 \oplus C_7$;

$R_{32} = C_1$;

.

$R_{162} = C_{14}$.

Соответствующий этой системе двухвходовой СА изображен на рис. 3.13.

Продолжив вычисления для последующих моментов времени, получим для момента времени t_8:

$R_{18} = D_8 \oplus C_9 \oplus C_5 \oplus C_2 \oplus R_{77}$;

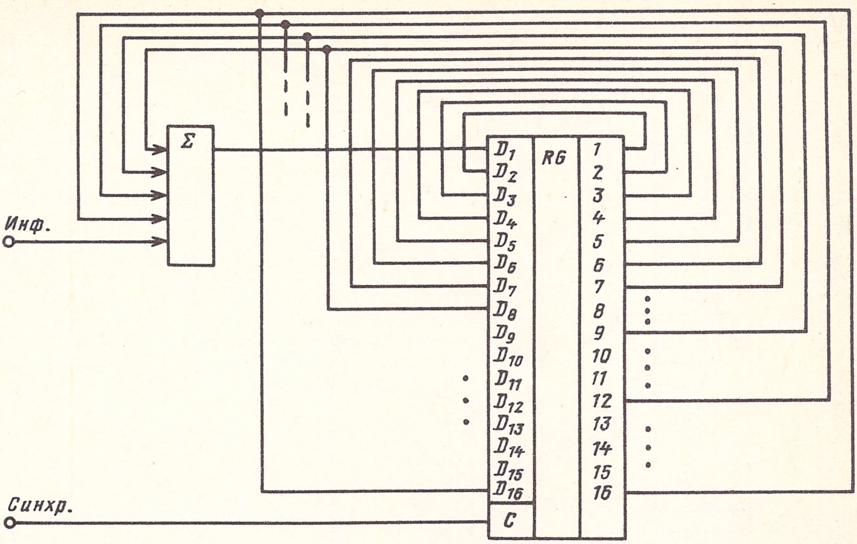


Рис. 3.12. Одновходовой СА на основе параллельного регистра и сумматора по модулю 2

$$R_{28} = D_7 \oplus C_{10} \oplus C_6 \oplus C_3 \oplus C_1;$$

$$R_{38} = D_6 \oplus C_{11} \oplus C_7 \oplus C_4 \oplus C_2;$$

$$R_{48} = D_5 \oplus C_{12} \oplus C_8 \oplus C_5 \oplus C_3;$$

$$R_{58} = D_4 \oplus C_{13} \oplus C_9 \oplus C_6 \oplus C_4;$$

$$R_{68} = D_3 \oplus C_{14} \oplus C_{10} \oplus C_7 \oplus C_5;$$

$$R_{78} = D_2 \oplus C_{15} \oplus C_{11} \oplus C_8 \oplus C_6;$$

$$R_{88} = D_1 \oplus C_{16} \oplus C_{12} \oplus C_9 \oplus C_7;$$

$$R_{98} = C_1;$$

.

$$R_{168} = C_8 .$$

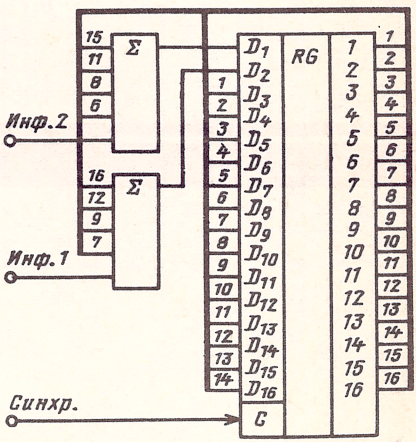


Рис. 3.13. Двухвходовой СА

Как видно, в структуре соответствующего СА (рис. 3.14) появляется дополнительная связь с выхода восьмого на вход первого сумматоров по модулю 2.

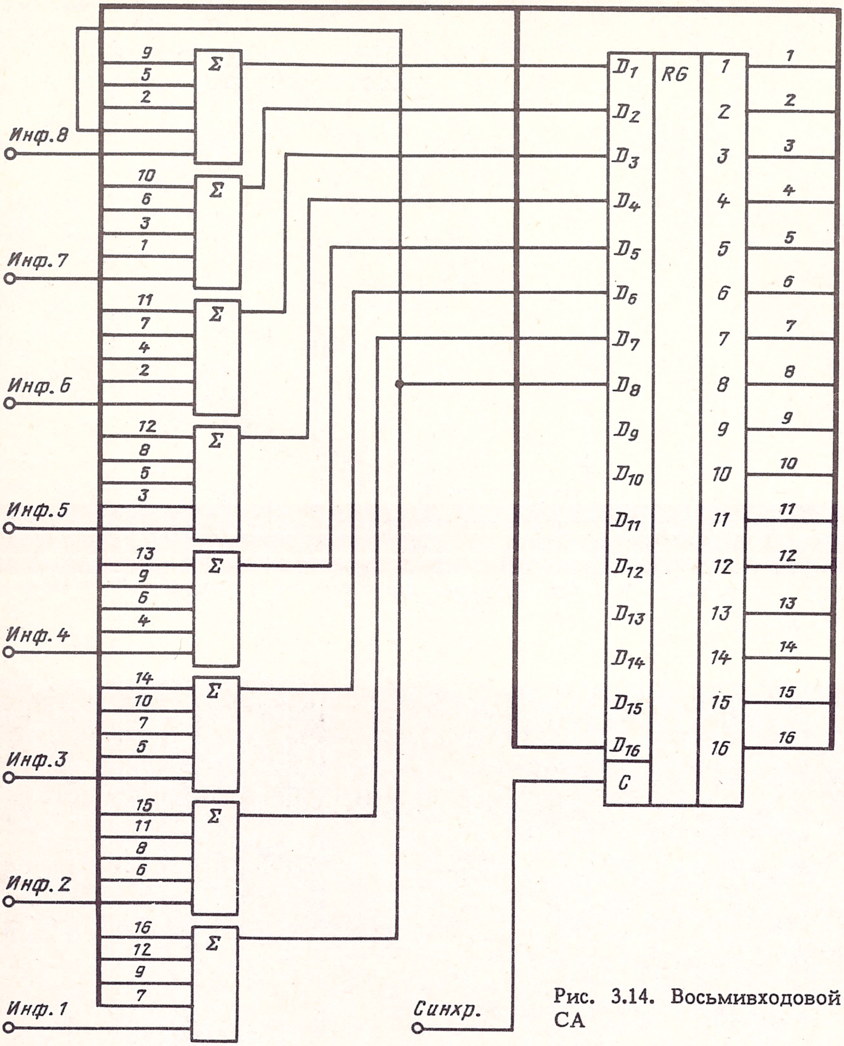


Рис. 3.14. Восьмивходовой СА

Продолжая вывод аналогичных систем уравнений последовательно для каждого из моментов времени, можно получить систему, описывающую ПСА с произвольным количеством информационных входов. Так, для $K = 18$ система будет иметь следующий вид:

$$R_{118} = D_{18} \oplus R_{1617} \oplus R_{1217} \oplus R_{917} \oplus R_{717};$$

$$R_{218} = D_{17} \oplus R_{1616} \oplus R_{1216} \oplus R_{916} \oplus R_{716};$$

$$R_{318} = D_{16} \oplus C_{1} \oplus R_{1215} \oplus R_{915} \oplus R_{715};$$

$$R_{418} = D_{15} \oplus C_{2} \oplus R_{1214} \oplus R_{914} \oplus R_{714};$$

$$R_{518} = D_{14} \oplus C_{3} \oplus R_{1213} \oplus R_{913} \oplus R_{713};$$

$$R_{618} = D_{13} \oplus C_{4} \oplus R_{1212} \oplus R_{912} \oplus R_{712};$$

$$R_{718} = D_{12} \oplus C_{5} \oplus C_{1} \oplus R_{911} \oplus R_{711};$$

$$R_{818} = D_{11} \oplus C_{6} \oplus C_{2} \oplus R_{910} \oplus R_{710};$$

$$R_{918} = D_{10} \oplus C_{7} \oplus C_{3} \oplus R_{99} \oplus R_{79};$$

$$R_{1018} = D_{9} \oplus C_{8} \oplus C_{4} \oplus C_{1} \oplus R_{78};$$

$$R_{1118} = D_{8} \oplus C_{9} \oplus C_{5} \oplus C_{2} \oplus R_{77};$$

$$R_{1218} = D_{7} \oplus C_{10} \oplus C_{6} \oplus C_{3} \oplus C_{1};$$

$$R_{1318} = D_{6} \oplus C_{11} \oplus C_{7} \oplus C_{4} \oplus C_{2};$$

$$R_{1418} = D_{5} \oplus C_{12} \oplus C_{8} \oplus C_{5} \oplus C_{3};$$

$$R_{1518} = D_{4} \oplus C_{13} \oplus C_{9} \oplus C_{6} \oplus C_{4};$$

$$R_{1618} = D_{3} \oplus C_{14} \oplus C_{10} \oplus C_{7} \oplus C_{5}.$$

Функциональная схема ПСА, описываемого данной системой уравнений, может быть построена аналогично описанному выше.

Являясь полной функциональной моделью последовательного СА, ПСА имеет одинаковые с ним суммарные характеристики по достоверности контроля, однако с точки зрения обнаружения ошибок вполне определенной конфигурации, т.е. в распределении классов необнаруживаемых ошибок, между последовательными СА (и параллельными СА с использованием мультиплексcра), с одной стороны, и ПСА, с другой стороны, имеются существенные различия, связанные с различием формирования эквивалентной входной информационной последовательности: ”вдоль” у последовательных СА и ”поперек” у ПСА.

Различия легко показать на примере двойных ошибок. Как известно, в последовательном СА двойная ошибка не

будет обнаружена, если она имеет вид: $\underbrace{100\ldots01}_{2^m-1}$, где m – разрядность образующего полинома. В то же время для отдельного входа ПСА разрядность необнаруживаемой двойной ошибки сокращается в K раз, где K – число информационных входов ПСА.

Более того, например, для 17-входового ПСА с образующим полиномом $P(X) = X^{16} + X^{12} + X^{9} + X^{7} + 1$ одновременные ошибки в произвольном такте по 1, 8, 10, 13 и 17-му выходам не будут обнаружены, что эквивалентно для одноканального СА последовательности на 1, 8, 10, 13 и 17-м тактах контроля. Последнее, очевидно, гораздо менее вероятно.

Для приведения характеристик достоверности ПСА в соответствие с характеристиками последовательных СА, используют, например, ОЗУ для запоминания промежуточных сигнатур.

Функциональная схема подобного СА приведена на рис. 3.15. В нем, в отличие от описанных ПСА, преобразование входного потока информации идет не "поперек", как в ПСА, а "вдоль", как в последовательных СА, ввиду чего они имеют идентичные характеристики по достоверности контроля.

Как уже говорилось, в случае неравенства числа выходов контролируемого устройства и числа входов ПСА на незадействованные входы подается детерминированная, незначащая информация. В рамках принятых ранее обозначений это эквивалентно подаче на соответствующий последовательный СА входной последовательности:

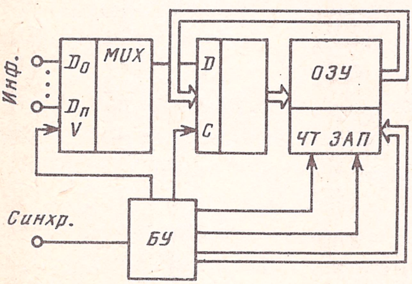


Рис. 3.15. СА с промежуточным запоминанием сигнатур

$$\overline{A = a_{11}\, a_{12} \, \ldots \, a_{1k} \underbrace{00 \, \ldots \, 0}_{m-k} a_{21}\, a_{22} \, \ldots}$$

$$\overline{\ldots \, a_{2k} \underbrace{00 \, \ldots \, 0}_{m-k} \ldots \, a_{n1}\, a_{n2} \, \ldots \, a_{nk} \underbrace{00 \, \ldots \, 0}_{m-k}} .$$

В этом случае пропорционально количеству незадействованных входов ПСА возрастает вероятность необнаружения ошибок малой кратности.

Для устранения данного недостатка применяются схемы ПСА с переменным числом информационных входов. ПСА, собранный по данной схеме, содержит регистр разрядностью m (где m – степень образующего полинома) и комбинационную схему, представляющую собой, по-существу, совокупность комбинационных схем, построенных по приведенному выше алгоритму, для различного числа информационных входов. Коммутация комбинационных схем осуществляется с помощью группы мультиплексоров.

Существенным недостатком данного ПСА является его чрезмерная сложность при аппаратной реализации. Однако для произвольного числа входов, не превышающих 16, его реализация в виде специализированной БИС, по-видимому, вполне реальна.

Большинство МСА характеризуется большими аппаратными затратами. В то же время для обеспечения, например, встроенного контроля, при котором требуется получать информацию со множества контрольных точек и не требуется производить поиск неисправностей, МСА являются наиболее эффективными для этой цели. Для подобных целей, когда жестко ограничиваются аппаратные затраты на реализацию МСА, может быть реализован простейший его вариант, схема которого приведена на рис. 3.16.

Этот МСА (по классификации его следует отнести к ПСА) по суммарной достоверности контроля по отношению к произвольным классам ошибок не отличается от любого другого СА, т.е. его достоверность зависит лишь от разрядности получаемой сигнатуры. В то же время по отношению к ряду ошибок малой кратности его обнаруживающая способность существенно хуже. Так, данным СА не будет обнаружена часть ошибок, возникших на соседних входах в двух смежных

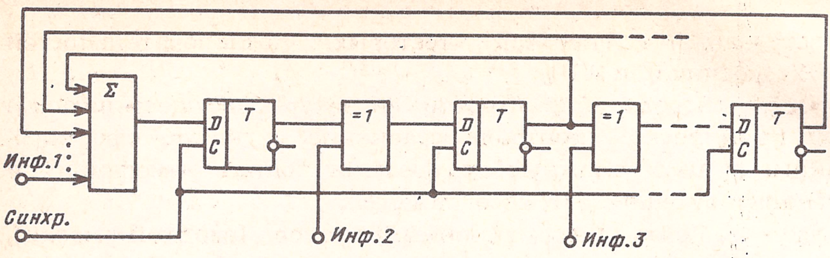


Рис. 3.16. Простейший вариант МСА

тактах синхронизации (в случае, если ни от одного из соответствующих триггеров нет отвода обратной связи на сумматор по модулю 2), не будут также обнаружены ошибки по входам N и $N+2$, происшедшие соответственно в момент времени t и $t+2$, (при условии, что ни один из триггеров N, $N+1$ и $N+2$ не имеет отвода обратной связи) и т.д.

Однако преимущество простоты этой схемы настолько велико, что она стала основой целого ряда встроенных систем функционального контроля на базе сигнатурного анализа (например, системы контроля BILBO – BIDCO и ряд других систем). Недостаток схемы, заключающийся в достоверности по отношению к ошибкам малой кратности, может быть преодолен рядом способов:

предварительным анализом контролируемого узла с точки зрения наиболее вероятных ошибок и соответствующим распределением контрольных точек между информационными входами анализатора;

введением дополнительных связей сумматоров по модулю 2 с тем, чтобы каждый очередной бит информации записывался не в один триггер, а в несколько (например, в два).

Все вышесказанное относилось, по существу, к формирователям сигнатур, которые являются основной частью сигнатурных анализаторов. Большинство реальных конструкций многовходовых, также как и последовательных, СА имеют ряд вспомогательных узлов для обеспечения более широких функциональных возможностей. К таким вспомогательным узлам относятся:

узел сравнения сигнатур с эталонными значениями, содержащий, как правило, счетчик и ПЗУ для хранения эталонных сигнатур;

встроенный генератор тестовых последовательностей (ПЗУ, счетчик или ГПЧ);

узел контроля стабильности сигнатуры, осуществляющий при циклическом контроле сравнение сигнатур, сформированных в двух смежных во времени "окнах измерения" и сигнализирующий об их несовпадении;

узел выбора информационных входов (входной маски), позволяющий подавать на вход те или иные сочетания выходных сигналов объекта контроля;

узел самоконтроля, позволяющий, как следует из названия, осуществлять контроль собственного функционирования СА как в автономном режиме, так и непосредственно в процессе функционирования;

узел адресации, запуска, останова, формирования "окна измерения", выбора фронтов входных сигналов и прочие узлы, отражающие конкретные условия применения СА. Все эти узлы являются общими для любых типов СА и поэтому в классификации не рассматриваются.

Таким образом, к основным достоинствам многовходовых СА следует отнести:

возможность функционального контроля цифровых узлов с большим количеством выходов;

высокое быстродействие за счет принципа параллельной обработки информации.

К недостаткам их относятся:

сложность аппаратной реализации;

сложность диагностирования неисправностей из-за необходимости локализации последней, как правило, с точностью до неисправного контакта.

Исходя из достоинств многовходовых СА применение их рекомендуется преимущественно для организации контроля ОТК и входного контроля готовых изделий – от микросхем до сложных цифровых узлов, а также для обеспечения встроенного контроля правильности функционирования узлов (при реализации простейших с точки зрения конструкции СА с целью экономии аппаратных затрат).

Для поиска неисправностей в цифровых узлах многовходовые СА, как правило, неприменимы, поскольку для локализации неисправности требуется разработка уточняющих таблиц эталонных сигнатур и переход к использованию многовходового СА в режиме одноканального.

Глава 4

МЕТОДЫ ОРГАНИЗАЦИИ СИСТЕМ ДИАГНОСТИРОВАНИЯ МПК С ИСПОЛЬЗОВАНИЕМ СИГНАТУРНОГО АНАЛИЗА

4.1. ОБЩИЕ ВОПРОСЫ

Задача организации системы диагностирования МПК с использованием сигнатурного анализа может рассматриваться как часть задачи организации систем диагностирования вычислительных систем и машин. Как любая вычислительная система, МПК обладает определенным уровнем эффективности, который обеспечивается на всех этапах его создания и эксплуатации. Под эффективностью здесь понимается степень соответствия МПК целевому назначению – решению различных задач вычислительного и управленческого характера [14].

Эффективность МПК снижается в процессе эксплуатации за счет ненадежности входящих в его состав элементов [14]. Эффективность МПК может быть повышена путем создания и использования средств обнаружения, локализации неисправностей и восстановления его работоспособности.

Комплексная система диагностирования – совокупность программных и аппаратных средств, предназначенных для автоматизированного или автоматического определения технического состояния МПК и поддержания необходимого уровня его эффективности [16, 66–69]. Комплексная система диагностирования решает следующие основные задачи:

обнаружение ошибки в работе МПК – задача контроля и профилактики;

определение характера ошибки – отказ или сбой;

локализация отказавшего элемента;

устранение ошибки, вызванной отказом или сбоем.

Под ошибкой здесь понимается всякое несоответствие между выполненным действием (операцией) и действием, которое должно быть выполнено в соответствии с исполняемым алгоритмом.

Для микропроцессорных комплектов интегральных схем, которые, как было показано в табл. 1.1, практически не имеют встроенных средств контроля, задача контроля явля-

ется одной из сложнейших задач, поскольку вся обработка информации в БИС практически не может быть проконтролирована без введения дублирования или специальных программных средств контроля, которые, в свою очередь, также будут недостаточно надежны, так как в случае отказа в микропроцессоре или в магистралях невозможно правильное выполнение программ [70].

Определение характера ошибки – отказ или сбой – при наличии системы контроля в МПК обычно производится путем повтора подпрограммы, во время которой возникла ошибка. При отсутствии системы контроля определение характера ошибки чаще всего производится совместно с локализацией отказавшего элемента.

Устранение ошибки для сложных МПК заключается в определении отказавшего блока или устройства МПК и последующей реконфигурации с целью восстановления его работоспособности без остановки на ремонт. Факт ошибки при этом должен фиксироваться и сообщаться обслуживающему персоналу. Непосредственный ремонт должен осуществляться после того, как в работе МПК настанет момент времени, когда возможна его остановка. В более простых МПК производится остановка работы и ремонт отказавшего устройства платы сразу после локализации отказа.

Система диагностирования МПК с использованием сигнатурного анализа решает в рамках комплексной системы диагностирования следующие задачи:

обнаружение ошибки в работе МПК при профилактике;

локализацию отказавшего элемента схемы.

Функцию обнаружения ошибки в работе МПК в режиме профилактики система диагностирования с использованием сигнатурного анализа выполняет в том случае, когда по тем или иным причинам в проверяемом МПК невозможно проведение профилактики с использованием функциональных тестовых проверок.

Таким образом, систему диагностирования МПК с использованием сигнатурного анализа можно определить как часть комплексной системы диагностирования МПК. Эта система диагностирования должна разрабатываться с учетом конкретных особенностей МПК.

Одним из основных отличий в организации систем диагностирования МПК от организации систем диагностирования

больших ЭВМ и УВМ является невозможность создания системы диагностирования МПК в организации, разрабатывающей устройства, из которых собирается МПК.

Микропроцессорные устройства и устройства расширения функций, как правило, выпускаются заводом-изготовителем в так называемой базовой конфигурации. Пользователь, разработчик МПК, закупает необходимый ему набор устройств, реконфигурирует их в соответствии с поставленной задачей и собирает МПК. Созданный пользователем МПК не является стандартным, он может быть уникальным, не имеющим аналогов или, если задача имеет не одно применение, таких МПК может быть создано некоторое множество. Следовательно, задача создания системы диагностирования распадается на несколько этапов и в общем случае решается различными организациями.

Организация, разрабатывающая микропроцессорные устройства, конструирует их с учетом требований сигнатурного анализа и выпускает систему поиска неисправности на базовую конфигурацию (БК) каждого из устройств. Задача создания системы диагностирования (СД) МПК с использованием сигнатурного анализа (САН) в общем случае решается пользователем (разработчиком МПК). Один из вариантов методики создания системы диагностирования МПК с использованием сигнатурного анализа представлен на рис. 4.1.

Разработка системы диагностирования существенно усложняется тем, что ее выполнение возлагается на организацию, разрабатывающую МПК, которая, как правило, не имеет опыта в создании систем диагностирования с использованием сигнатурного анализа. Более того, ввиду небольшого числа производимых МПК (иногда он создается вообще в единственном числе) разрабатывающая организация часто не видит необходимости в выделении сил и средств на эту разработку. Поэтому одним из главных условий успешного внедрения систем диагностирования МПК с использованием сигнатурного анализа является построение такой методики, которая существенно упростила бы эту процедуру и свела бы к минимуму затраты на создание системы диагностирования. Для этого организация, разрабатывающая микропроцессорные устройства вместе с системой диаг-

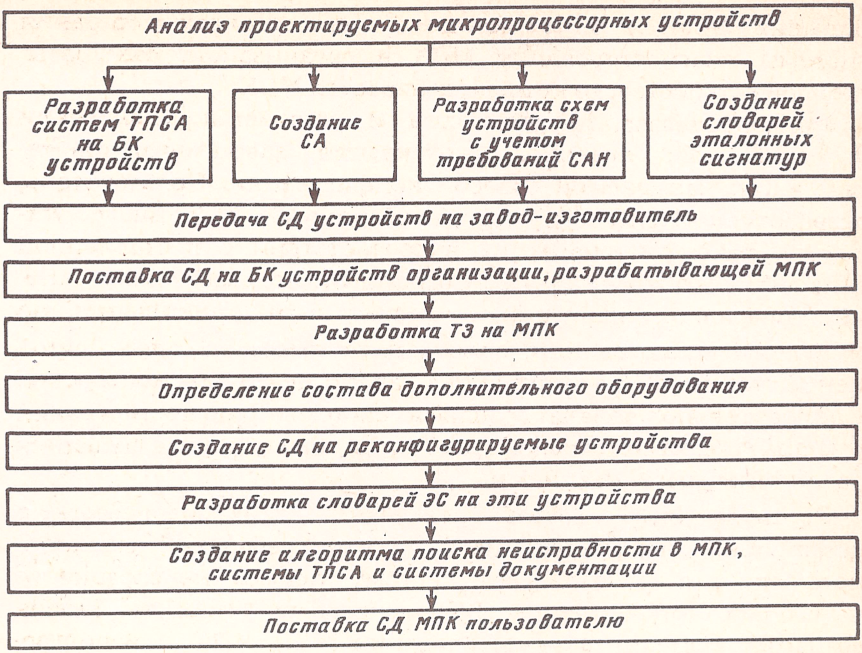


Рис. 4.1. Методика создания системы диагностирования МПК с использованием сигнатурного анализа

ностирования на базовую конфигурацию, должна выпустить набор алгоритмов поиска неисправности с использованием сигнатурного анализа для тех конфигураций, которые могут быть созданы на базе этих устройств. Система диагностирования на каждое устройство должна быть по возможности создана таким образом, чтобы при его реконфигурации для получения новых эталонных сигнатур (ЭС) достаточно было бы снять эти сигнатуры с реконфигурированного устройства по рекомендациям разработчика устройства.

С учетом требований к составу системы диагностирования МПК, изложенных в первой главе, состав системы диагностирования МПК с использованием сигнатурного анализа может быть следующим:

1) устройство сигнатурного анализа;

2) дополнительная аппаратура, вводимая в МПК и в устройства, входящие в его состав, которая необходима для ор-

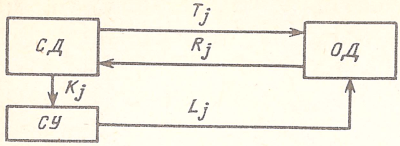


Рис. 4.2. Обобщенная схема тестового диагностирования

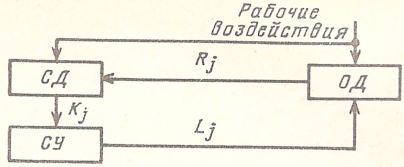


Рис. 4.3. Обобщенная схема функционального тестирования

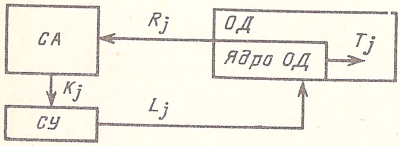


Рис. 4.4. Вариант схемы диагностирования с использованием метода сигнатурного анализа

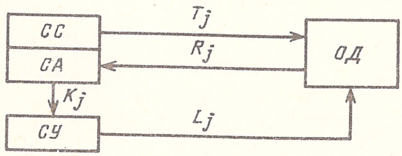


Рис. 4.5. Вариант схемы диагностирования с использованием сигнатурного анализа со стимулятором

ганизации поиска неисправности с использованием сигнатурного анализа;

3) система контрольных точек;

4) алгоритмы проверки МПК и устройств, входящих в его состав;

5) система тестовых последовательностей сигнатурного анализа;

6) система документации, необходимой для поиска неисправности.

Обобщенные схемы тестового и функционального диагностирования технических объектов рассматривались в [14, 66, 69] (рис. 4.2, 4.3), где *ОД* – объект диагностирования; *СД* – средства диагностирования; *СУ* – средства устранения неисправностей; K_j – схема с локализованной неисправностью; L_j – отремонтированная схема; T_j – подаваемое в ОД тестовое воздействие; R_j – реакция ОД на T_j, поступающая в СД.

Аналогичным образом можно определить схемы диагностирования МПК с использованием сигнатурного анализа (рис. 4.4, 4.5), где *СА* – сигнатурный анализатор; *СС* – средства генерации стимулирующих воздействий.

На рис. 4.4 изображен вариант схемы диагностирования с использованием сигнатурных анализаторов первого поколения типа HP5004A, СЛО1П [25, 30]. На рис. 4.5 показан вариант схемы диагностирования с использованием внешних средств генерации стимулирующих воздействий на проверяемый МПК. Могут использоваться СС как совмещенные с сигнатурным анализатором, например 7201 Locator [8], так и раздельные, например стимуляторы HP5001A, HP5001C, HP5001 [71–73], которые используются совместно с сигнатурными анализаторами HP5005A, HP5005B [75, 76].

Уже из этих укрупненных схем видно, что в рамках создания системы диагностирования необходимо решить такие задачи, как выделение оптимального ядра в МПК, определение способов его диагностирования, разработка алгоритма поиска неисправности и т.д. Разрабатываемая система диагностирования должна быть рациональной.

Под рациональной системой диагностирования МПК с использованием метода сигнатурного анализа понимается система диагностирования, позволяющая производить поиск неисправности с требуемой глубиной локализации L за минимальное время τ при одновременном уменьшении количества дополнительного оборудования D, необходимого для ее реализации, при сокращении объема документации V и снижении требований к уровню квалификации обслуживающего персонала H.

Создание такой системы диагностирования представляет собой итерационный процесс, который может быть описан в виде следующей последовательности:

1) выбор параметров устройства сигнатурного анализа;

2) определение состава дополнительного оборудования, необходимого для реализации метода сигнатурного анализа;

3) выбор контрольных точек;

4) разработка алгоритмов проверки МПК и устройств, которые входят в его состав;

5) создание систем ТПСА;

6) разработка документации;

7) оценка полученной системы диагностирования и ее модификации согласно п. 1–6 описанного процесса до тех пор, пока не будут удовлетворены требования к параметрам L, τ, D, V, H.

На каждом из перечисленных этапов для выбора рациональных структур привлекаются как количественные, так и качественные оценки.

4.2. ИССЛЕДОВАНИЕ МЕТОДОВ И АЛГОРИТМОВ ПРОВЕРКИ МПК

Выбор методов и алгоритмов проверки МПК с использованием сигнатурного анализа существенно зависит от состава и структуры МПК.

Устройства, входящие в состав МПК, могут быть как одноплатными, так и многоплатными. При сборке МПК они устанавливаются в панель блока МПК, объединяющую их с помощью межмашинной магистрали (ММ), которая состоит из шин адреса, данных и управления. Для объединения многоплатных устройств в одно целое используются дополнительные разъемы, расположенные, например, на панели блока МПК, на которых каким-то индивидуальным образом (к примеру, с помощью накрутки) образуются необходимые межплатные связи конкретного устройства.

Кроме перечисленных устройств в МПК могут входить: внешние запоминающие устройства (ВЗУ) – накопители на гибких или жестких магнитных дисках, кассетные магнитофоны, а также устройства управления и отображения (дисплеи, пульты управления и т.д.)

Естественно, что может существовать огромное количество различных МПК, отличающихся друг от друга в той или иной своей части. Однако большинство из них может быть получено при помощи различных комбинаций из следующих четырех типов МПК:

однопроцессорный МПК без средств связи с оператором и без внешнего запоминающего устройства;

однопроцессорный МПК со средствами связи с оператором (дисплей, телетайп, пульт и т.д.) и с ВЗУ (кассетный магнитофон, дисковое ЗУ и т.д.);

многопроцессорный МПК без средств связи с оператором, не имеющим ВЗУ и средств контроля;

многопроцессорный МПК со средствами связи с оператором, имеющих ВЗУ и средства контроля.

Перечисленные выше типы МПК в общем случае могут служить в качестве моделей для создания обобщенных ал-

горитмов проверки МПК с помощью сигнатурного анализа, поскольку наличие или отсутствие ВЗУ или (и) средств связи с оператором существенно влияет на выбор алгоритма проверки, а также на способ хранения системы тестовых последовательностей сигнатурного анализа и способ их инициализации.

Поиск неисправности в МПК с использованием сигнатурного анализа может производиться по одному из двух алгоритмов:

1) поиск неисправности в МПК с использованием только сигнатурного анализа;

2) с предварительным определением неисправного устройства с помощью функциональных тестов и последующей локализацией неисправной ИС с использованием сигнатурного анализа.

Выбор одного из этих алгоритмов определяется наличием функциональных тестов в данном МПК и возможностью их запуска. В МПК, например, могут не входить дисплей или телетайп, необходимые для запуска теста, или возникшая неисправность такова, что запуск тестов невозможен. В некоторых случаях возможно использование комбинации из этих двух алгоритмов.

Алгоритм поиска неисправности в МПК с использованием только сигнатурного анализа представлен на рис. 4.6. Алгоритм реализует метод раскрутки в применении к сигнатурному анализу.

Проверка начинается с центрального процессорного устройства. После его проверки и, если необходимо, ремонта обнаруженной неисправности в ОЗУ устройства с внешнего носителя вводится тестовая программа для проверки ММ и плат, подсоединенных к ней.

Проверка ММ производится при выключенных формирователях шин, которые осуществляют развязку ММ и внутренней магистрали (ВМ) каждой платы во всех устройствах, входящих в МПК, кроме проверенного ЦПУ. Эта проверка позволяет убедиться в том, что в ММ нет коротких замыканий, что обмен информацией между внутренней магистралью ЦПУ и ММ проходит нормально.

После проверки ММ производится последовательная проверка всех устройств, входящих в состав МПК.

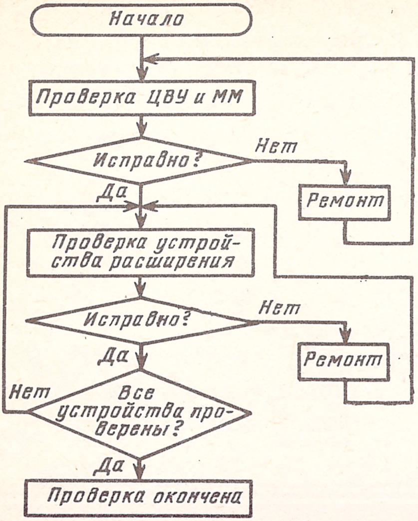


Рис. 4.6. Блок-схема алгоритма поиска неисправности в МПК с помощью только сигнатурного анализа

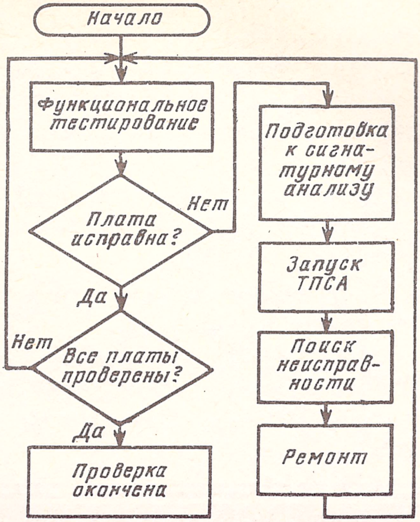


Рис. 4.7. Блок-схема алгоритма поиска неисправности в МПК

После обнаружения неисправности и ее ремонта производится повторная проверка МПК.

Алгоритм поиска неисправности с предварительным определением неисправного устройства с помощью функциональных тестов и последующей локализацией неисправности с использованием сигнатурного анализа представлен на рис. 4.7.

Проверка начинается с запуска функциональных тестов устройства. После определения неисправного устройства или группы устройств, что зависит от разрешающей способности функциональных тестов, производится подключение СА и поиск неисправности. После ремонта обнаруженной неисправности производится повторный запуск функциональных тестов для проверки правильности функционирования МПК.

Анализируя оба рассмотренных алгоритма поиска неисправности в МПК с использованием сигнатурного анализа, видим, что их основой является алгоритм поиска неисправностей в устройствах.

В данной книге рассматриваются ЦПУ, содержащие в своем составе ОЗУ, ПЗУ, устройство ввода-вывода, обеспечивающее связь с каким-нибудь ВЗУ, т.е. ЦПУ, которое, как показано в первой главе, является диагностирующей подсистемой. Если же ЦПУ не содержит одного или нескольких из перечисленных выше устройств, что делает его недиагностирующей подсистемой, то система поиска неисправности в МПК, созданном на базе таких ЦПУ, может быть аналогична системе диагностирования ЦПУ, являющегося диагностирующей подсистемой, которая рассматривается ниже. Возможность подобного сопоставления систем поиска неисправности объясняется практически одинаковой структурой шин внутриустройственной и межмашинной магистралей.

В рассмотренных алгоритмах предполагается также, что часть ТПСА может быть размещена в ПЗУ ЦПУ, а оставшиеся ТПСА вводятся в его ОЗУ с ВЗУ. Однако возможны и другие варианты, например, в ПЗУ может храниться весь объем ТПСА или стимуляция схем может осуществляться при помощи ТПСА, задаваемых через внешний стимулятор.

В общем случае в МПК может существовать некоторое множество ЦПУ, устройств расширения памяти и вообще некоторое множество устройств каждого типа. Основными элементами, обеспечивающими существование МПК, являются ММ и интерфейсные схемы, через которые устройства, входящие в состав МПК, подсоединяются к ММ. Характер связей и структура ММ, как правило, не зависят от количества и состава устройств, подсоединяемых к ней. Для разработки алгоритма проверки МПК необходимо определить порядок и правила проверки ММ. Для формализации воспользуемся представлением МПК в виде графа. Выделим в графе $G(E^1, U^1)$ (см. рис. 1.8) подграф $G^2(Q, U^2)$, вершинами $q_i \in Q$ (где $i=\overline{1,n}$) которого являются интерфейсные схемы всех устройств МПК, за исключением ММ, а дугами $U_{ij} \in U^2$ – связи между ними, причем $U^2 \in U^1$. В графе G^2 каждая вершина связана со всеми остальными, что вытекает из характера связей в ММ.

Полученный $G^2(Q, U^2)$ является полным, симметрическим графом. Поскольку граф $G^2(Q, U^2)$ – полный, то в нем нельзя

выделить некоторое множество вершин $S \subset Q$, которое будет внутренне устойчивым, так как по определению множество S будет внутренне устойчивым, если никакие две вершины из S не смежны [49]. Из этого следует, что граф $G^2(Q, U^2)$ не имеет ядра. Следовательно, в ММ невозможно выделить ядро оборудования, с помощью которого можно было бы осуществить проверку остальных интерфейсных схем, подсоединенных к магистрали.

Исключим из графа $G^2(Q, U^2)$ все вершины, кроме q_j (где $j = \overline{1,f}$), принадлежащей одному из ЦПУ, которое, как показано в первой главе, является диагностирующей подсистемой. Полученный подграф $G_1^2(Q_1, U_1^2)$ имеет одну вершину q_j^2, представляющую собой схему связи с ММ j-го ЦПУ и систему шин ММ.

Введем в подграф $G_1^2(Q_1, U_1^2)$ дополнительную вершину q_{CA}, не принадлежащую множеству Q_1, связанную с q_j дугой W_{jCA}. Вершина q_{CA} и дуга W_{jCA} соответственно представляют собой СА и связь, по которой в СА поступают управляющие и информационные сигналы. Исследуем полученный граф $G(Q_2, W)$ на наличие ядра. Поскольку в данном графе множество вершин $q_\theta \in Q_2$ состоит из вершин $q_j \in Q_1$ и $q_{CA} \in Q_2$, причем $Q_1 \in Q_2$ и $q_{CA} \notin Q_1$, то можно записать следующие соотношения [77]:

$$(\forall\, q_\theta \in Q_1)\; Q_1 \cap \Gamma q_\theta = \phi, \tag{4.1}$$

$$(\forall\, q_\theta \notin Q_1)\; Q_1 \cap \Gamma q_\theta \neq \phi. \tag{4.2}$$

Из (4.1) и (4.2) следует, что множество Q_1 одновременно как внутренне, так и внешне устойчивое, т.е. множество Q_1 является ядром графа $G(Q_2, W)$. Таким образом, вершина q_j может быть использована (после ее проверки) в качестве ядра при проверке остальных вершин графа $G^2(Q, U^2)$, рассматриваемых как внешние по отношению к Q_2 вершины. Образуем граф $G_1(Q_3, W_2)$ добавлением к графу $G(Q_2, W)$ вершины $\beta_i \in Q$, $i = \overline{1,n}$ и слиянием вершин q_j и q_{CA}. Произведем проверку q_i. Процесс продолжим до тех пор, пока не будут провере-

ны все вершины $q_i \in Q$. Из этих рассуждений следует, что одним из возможных алгоритмов поиска неисправности в ММ является постепенное наращивание проверенного оборудования с одновременным использованием СА.

Исследуемый процесс является неопределенным, поскольку не определен порядок подсоединения вершин q_i к ядру, т.е. не определена последовательность проверки устройств. Последовательность проверки может быть определена путем оценки зависимостей между устройствами МПК. Сформулируем следующее правило: i-е устройство всегда проверяется перед j-м, если оно используется для проверки j-го устройства. Анализ зависимостей между устройствами внутри МПК произведем с использованием графовой модели МПК. Поскольку каждой вершине $q_i \in Q$ однозначно соответствует вершина $e_i \in E^1$, то можно утверждать, что проведенные рассуждения не изменятся при объединении вершин q_i и e_i в одну вершину.

Рассмотрим граф $G(E^1, V)$, представленный в виде матрицы смежностей. Вершинами графа являются устройства $e^1_i \in E^1$, $e^1_i = e_i \cap q_i$, $\forall_i = \overline{1,n}$. Вершины e_i и e_j соединяются дугой V_{ij}, если устройство e_j проверяет или используется для проверки e_j, в этом случае представляется единица в клетке, образованной пересечением j-го столбца и i-й строки матрицы. Полученную матрицу (табл. (4.1) можно использовать для определения порядка проверки устройств в МПК. Предположим, что этот порядок является функцией сформированного выше правила проверки и порядкового номера вершины, причем при равенстве весовых коэффициентов первой проверяется вершина с меньшим порядковым номером. Весовой коэффициент представляет собой сумму чисел, полученных в столбцах транзитивного замыкания для данной вершины.

Для табл. 4.1 весовые коэффициенты будут равны:

$$\Sigma e_1 = 0,\ \Sigma e_2 = 1,\ \Sigma e_k = 3,\ \Sigma e_n = 2.$$

Следовательно, для данного примера последовательность проверки будет такова: первым проверяется устройство, соответствующее e_1, затем e_2, e_n и, наконец, e_k. Названия устройств подставляются в алгоритмы проверки МПК

Таблица 4.1

	e_1	e_2	…	e_k	…	e_n	$\hat{\Gamma} e_1$	$\hat{\Gamma} e_2$	$\hat{\Gamma} e_k$	$\hat{\Gamma} e_n$
e_1		1	…	1	…	1	0			
e_2			…		…	1	1	0		
.	.	.	…	.	…	.	.	.	.	.
.	.	.	…	.	…	.	.	.	.	.
.	.	.	…	.	…	.	.	.	.	.
e_k			…		…		1	2	0	1
.	.	.	…	.	…	.	.	.	.	.
.	.	.	…	.	…	.	.	.	.	.
.	.	.	…	.	…	.	.	.	.	.
e_n			…	1	…		1			0

(см. рис. 4.6 и рис. 4.7). Таким образом, может быть сформулирована формальная методика составления алгоритма проверки МПК с использованием сигнатурного анализа:

1. Строится квадратная матрица, строки и столбцы которой представляют собой множество E_1 вершин графа $G(E^1, V)$. Единица в клетку e_i, e_j становится в том случае, если e_i используется для проверки e_j.

2. Выбирается вершина e_i, принадлежащая первому ЦПУ, определяется ее транзитивное замыкание $\hat{\Gamma} e_i$.

3. Повторяется п. 2 для e_j (где $j = \overline{2,f}$) до тех пор, пока не будут вычислены $\hat{\Gamma} e_j$ для всех ЦПУ.

4. Определяется одна из вершин e_1, e_j, имеющая наибольшее $\hat{\Gamma} e_j$, т.е. связанная с наибольшим количеством вершин графа.

5. Из столбца $\hat{\Gamma} e_j$ выбранной вершины выбирается вершина $e_i \in E^1$, имеющая в нем наименьшее число.

6. Определяется $\hat{\Gamma} e_i$.

7. Пункты 5, 6 повторяются до тех пор, пока не будут вычислены $\hat{Г}e_i$ всех вершин, которые имеют ненулевые числа в $\hat{Г}e_j$.

8. Вычисляется весовой коэффициент всех вершин, для чего производится сложение чисел в $\hat{Г}e_j$ и во всех $\hat{Г}e_i$ для данной вершины.

9. Выписываются вершины в порядке возрастания весовых коэффициентов, а в случае их равенства – по возрастанию порядковых номеров.

10. Повторяются п. 4–9 до тех пор, пока не будет исчерпано все множество вершин E^1, т.е. все устройства МПК.

11. Названия устройств в порядке, который определен по п. 9 данной методики, подставляются в выбранный в соответствии с типом МПК алгоритм проверки. В результате подстановки определяется алгоритм проверки конкретного МПК.

Полученный алгоритм в основном совпадает с методом расширяющихся областей, который модифицируется с учетом специфики микропроцессорных устройств и применения метода сигнатурного анализа. Исключение устройств из ММ и последующее их поочередное подключение может осуществляться как при помощи их физического отсоединения от разъемов ММ, так и с помощью электрического отключения, в этом случае при проектировании МПК необходимо предусмотреть в устройствах ММ дополнительные блокирующие схемы.

4.3. АЛГОРИТМЫ ПРОВЕРКИ УСТРОЙСТВ, ВХОДЯЩИХ В СОСТАВ МПК. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К СОСТАВУ ДОПОЛНИТЕЛЬНОГО ОБОРУДОВАНИЯ

Алгоритмы поиска неисправности в устройствах зависят от их структуры, типа БИС, на базе которых выполнены, и способа стимуляции схем, входящих в состав устройства.

Сигнатурный анализ, как и большинство методов диагностики, требует введения в проверяемое устройство дополнительного оборудования, которое позволит выполнить требования, необходимые для реализации метода сигнатур-

ного анализа в проверяемом устройстве [25]. К этим требованиям относятся:

осуществление аппаратной или программной стимуляции схем проверяемого устройства;

цикличность стимуляции;

синхронность каждого проверяемого сигнала схемы с синхронизирующим импульсом, который используется в качестве СИ для СА;

наличие в проверяемом устройстве или МПК по крайней мере одного сигнала, синхронного с СИ, имеющего хотя бы один фронт или срез внутри цикла ТПСА, с помощью которого можно задать ОИ внутри стимулирующей последовательности;

постоянная длительность ОИ (постоянное количество фронтов и (или) спадов СИ внутри ОИ);

одинаковое изменение каждого проверяемого сигнала внутри двух последовательных во времени ОИ;

возможность разрыва обратных связей с целью предотвращения прохождения ошибки по цепям обратной связи и снижения из-за этого точности локализации;

возможность выборочного подключения устройств к магистрали для реализации метода раскрутки.

Анализ схемы устройства с целью определения ее соответствия требованиям сигнатурного анализа и определения состава дополнительного оборудования производится на этапе разработки каждого устройства. Анализ может производиться как вручную, путем непосредственного просмотра схем устройства, так и автоматически, путем анализа математической модели проверяемого устройства. Математическая модель может быть построена с использованием различных подходов, например: путем формализованного описания устройства на базе аппарата многоосновных алгебр [41] или при помощи создания его графовой модели.

Во время анализа схемы в первую очередь производится выделение кольцевых структур – схем с обратной связью. При этом элементы, осуществляющие двунаправленную передачу информации, также считаются схемами с обратной связью. Наиболее просто выделение кольцевых структур производится на графовой модели.

Представим схему устройства в виде направленного графа $G(E, V)$, где вершинам $e_i \in E$ соответствуют элементы схемы, а множеству дуг графа $u_{ij} \subset U$ – связи между вершинами e_i и e_j. Определим отображение Γ [76] следующим образом:

$$e_j \in \Gamma e_i, \tag{4.3}$$

если существует дуга u_{ij}. Таким образом, отображение Γ определяет множество дуг графа $G(E, U)$ и поэтому его можно записать как $G(E, \Gamma)$.

Задачу выделения кольцевых структур можно сформулировать на языке теории графов как разложение графа $G(E, \Gamma)$ на максимально сильносвязные подграфы $G'(E', \Gamma')$, где $E' \in E$ и представляют собой вершины $e_j \in E$, входящие в кольцевую структуру, а $\hat{\Gamma} e_j$ определяет множество дуг графа $G'(E', \Gamma')$. Поставленную задачу можно решить известным методом разложения графа на сильносвязные подграфы [77], который заключается в отыскании таких классов $c(e_j)$, содержащих e_j, что

$$c(e_j) = \hat{\Gamma} e_j \cap \hat{\Gamma}^{-} e_j. \tag{4.4}$$

Определим формализованный алгоритм поиска кольцевых структур, используя метод Мальгранжа [77]:

1. Запишем граф $G(E, \Gamma)$ в виде матрицы связностей.

2. Выберем произвольную вершину e_j графа $G(E, \Gamma)$, определим для нее прямое и обратное транзитивные замыкания $\hat{\Gamma} e_j$ и $\hat{\Gamma}^{-} e_j$.

3. Найдем пересечение $\hat{\Gamma} e_j$ и $\hat{\Gamma}^{-} e_j$. Полученное множество вершин $c(e_j)$ образует максимально сильносвязный подграф $G''(E, \Gamma)$, в который входит e_j.

4. Вершины $e_j \in c(e_j)$ выписываются, затем объединяются в одну вершину с сохранением всех внешних связей. В результате получается подграф $G'''(E', \Gamma)$.

5. Повторяются п. 2–4 до тех пор, пока множество вершин не будет просмотрено полностью.

При машинном анализе матрицы можно не определять длину пути из i-й вершины в j-ю, а при его наличии ставить в соответствующую клетку строки или столбца 1. Единицей же можно помечать выбранную вершину. В клетки строки

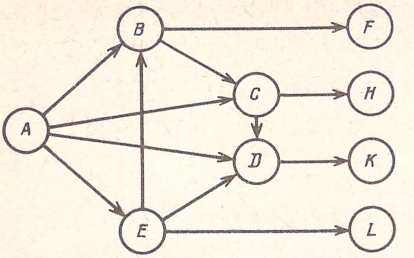


Рис. 4.8. Графовая модель упрощенной схемы регенерации информации в ОЗУ

	A	B	C	D	E	F	H	K	L	ГВ
A		1	1	1	1					
B			1			1				0
C				1			1			1
D					1			1		2
E		1							1	3
F										4
H										2
K										3
L										4
	1	0	3	2	1					

Рис. 4.9. Матрица смежностей для графа на рис. 4.8

или столбца, которые не имеют связи с выбранной вершиной, записать 0. Тогда операцию получения пересечения можно выполнить с помощью операции логического умножения, т.е.

$$c(e_i) = \hat{\Gamma} e_i \cap \hat{\Gamma}^- e_i = \hat{\Gamma} e_i \hat{\Gamma}^- e_i. \qquad (4.5)$$

Проиллюстрируем алгоритм поиска кольцевых структур на примере графа $G(E, \Gamma)$, представленного на рис. 4.8. Граф представляет собой упрощенную графовую модель схемы регенерации информации в оперативном запоминающем устройстве динамического типа. Вершина A отождествляется с выходом генератора тактовых импульсов, вершины B, C, D, E образуют счетчик регенерации, вершины F, H, K, L – коммутаторы, предназначенные для формирования адреса ОЗУ.

Для данного графа составим матрицу (рис. 4.9). Справа от матрицы нарисуем столбец, а снизу – строку, которые заполняются по следующему алгоритму:

в клетку столбца напротив строки выбранной вершины ставим 0; в клетки столбца, расположенные напротив строк, в которых строка выбранной вершины содержит 1, ставим 1; 2 – записывается в тех клетках столбца, которые расположены напротив строк, связанных с выбранной вершиной через одну вершину, помеченную 1, и т.д. Таким образом, столбец представляет собой транзитивное замыкание для выбранной

вершины $\hat{\Gamma} e_j$. Аналогично, заменяя строки на столбцы и и столбцы на строки, заполняем строку, которая представляет собой обратное транзитивное замыкание $\hat{\Gamma}^- e_j$.

На рис. 4.9 показано заполнение дополнительной строки и столбца для вершины B. Вершина A не рассматривается, поскольку она не входит в кольцевую структуру и, следовательно, $c(A) = 0$. В результате заполнения столбца и строки получим транзитивное замыкание $\hat{\Gamma} B$ и обратное транзитивное замыкание $\hat{\Gamma}^- B$, которые соответственно равны:

$$\hat{\Gamma} B = \{ B, C, D, E, F, H, K, L \}; \tag{4.6}$$

$$\hat{\Gamma}^- B = \{ A, B, C, D, E \}. \tag{4.7}$$

Определим пересечение выражений (4.6) и (4.7):

$$c(B) = \hat{\Gamma} B \cap \hat{\Gamma}^- B = \{ B, C, D, E \}. \tag{4.8}$$

Полученное множество вершин представляют собой счетчик регенерации памяти. После определения класса $c(B)$ вершины, входящие в состав класса, сводятся в одну вершину B_1. Затем аналогично анализируется следующая вершина, например F. Процедура продолжается для всех вершин.

В МПК, созданных на базе микропроцессорных комплектов ИС с однокристальным микропроцессором, наиболее важной обратной связью является двунаправленная магистраль данных (рис. 4.10), неисправность на одной из шин которой делает невозможным выполнение любой программы. Задача разрыва этой обратной связи и хотя бы частичной проверки микропроцессора является одной из основных задач проверки микропроцессорного устройства. Ей посвящено значительное число публикаций [26, 70, 78–82], где предлагаются различные варианты решения, которые можно разделить на две основные группы:

механический разрыв магистрали данных и такая коммутация сигналов на шине данных микропроцессора, которая подает на его вход одну и ту же команду, позволяющую осуществить непрерывное приращение счетчика команд микропроцессора;

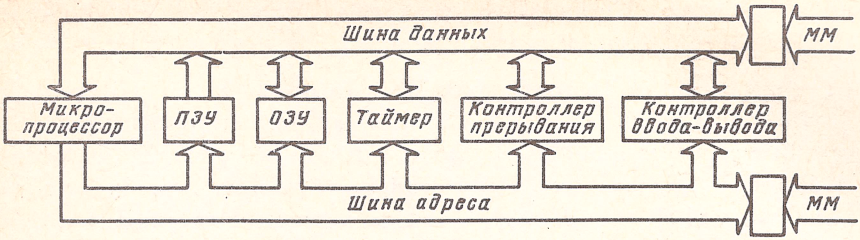


Рис. 4.10. Пример микропроцессорного устройства

электрическая коммутация сигналов блокировки, которая позволяет с помощью дополнительных схем, включенных в устройство, добиться той же цели.

Для осуществления механического разрыва магистрали данных и последующей коммутации сигналов на шине данных микропроцессора используются следующие способы:

установка на шине данных микропроцессора переключателей, с помощью которых шина данных микропроцессоров во время поиска неисправности отключается от остальных схем и на входы данных микропроцессора подается определенный код, например, как показано на рис. 4.11 [80];

установка шинных формирователей, развязывающих микропроцессор от остальных устройств, на соединители с целью их замены во время проверки на коммутирующие элементы, реализующие функции разрыва магистрали и подачи на шины данных микропроцессора требуемого кода команды, или, что одно и то же, введение на шине данных микропроцессора привязки к шине питания и коммутация некоторых из шин с помощью переключателей при снятых формирователях шин (рис. 4.12) [78, 82].

При электрической коммутации сигналов блокировки используются такие способы, как:

установка на шине данных микропроцессора дополнительных шинных формирователей, на входы которых подается код требуемой команды (дополнительные шинные формирователи включаются во время проверки одновременно с выключением основных шинных формирователей (рис. 4.13) [77];

принудительное подключение на время проверки межмашинной магистрали к шине данных микропроцессора,

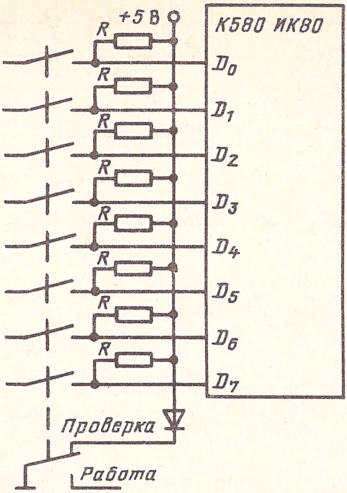


Рис. 4.11. Перевод микропроцессора в автономный режим с помощью переключателей

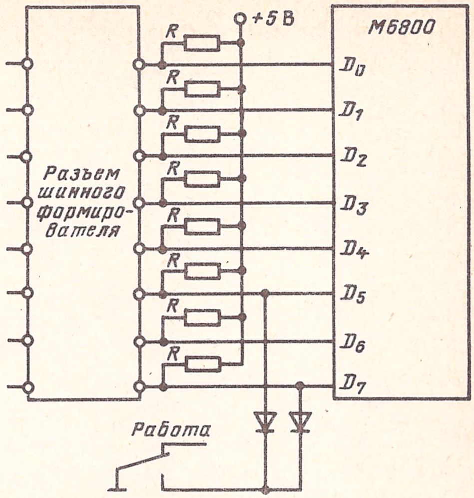


Рис. 4.12. Перевод микропроцессора в атомный режим работы с помощью коммутирующего элемента

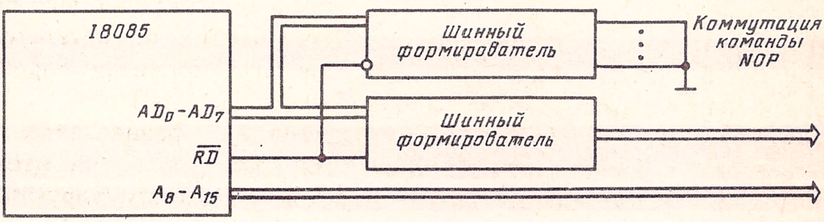


Рис. 4.13. Перевод микропроцессора в автономный режим работы с помощью дополнительного шинного формирователя

которое производится с помощью дополнительной схемы ИЛИ, позволяющей подать сигнал "Вкл. D" для включения инвертирующих шинных формирователей (рис. 4.14) [70].

Механические способы коммутации магистрали данных осуществляют в отличие от электрических способов гарантированное разъединение магистралей и подачу на микропроцессор требуемого кода. Однако их использование приводит к некоторому снижению надежности устройства по сравнению с электрическими способами, так как механические переключатели обладают меньшей, чем интегральные схемы,

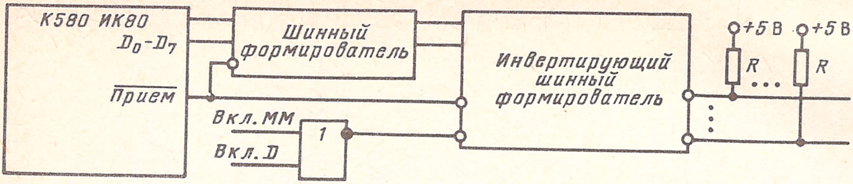


Рис. 4.14. Перевод микропроцессора в автономный режим работы путем подключения к ММ

надежностью, а при замене шинных формирователей надежность устройства снижается за счет введения соединителей под шинные формирователи и их быстрого износа. В электрических способах, которые значительно меньше снижают надежность устройства, все-таки существует возможность подачи на вход микропроцессора не того кода команды из-за отказов шинных формирователей.

Выбор того или иного способа разрыва магистрали данных зависит от конкретного исполнения центрального процессорного устройства. Причем способ электрической коммутации сигнала является предпочтительным, особенно в связи с тем, что в настоящее время появилась возможность проверки отдельных подозреваемых ИС, например, с помощью прибора Щ4315 или индивидуальных внутрисхемных тестеров.

Для микропроцессорных устройств, создаваемых на базе микропроцессорных секций с микропрограммным управлением (рис. 4.15), хорошим способом проверки является введение дополнительного счетчика, который используется для задания адреса микрокоманд [70, 83].

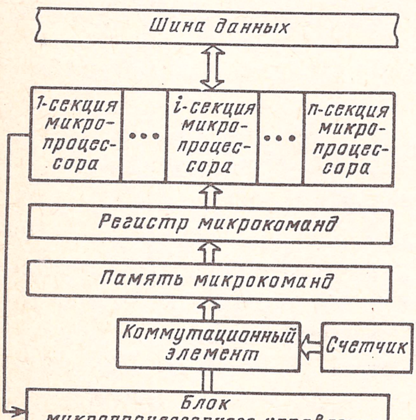


Рис. 4.15. Микропроцессорное устройство на базе секционированного микропроцессора с дополнительным оборудованием для автономной стимуляции

Другие схемы с обратными связями можно разделить на три основные группы: выполненные на одном корпусе ИС; устанавливаемые в исходное состояние; не устанавливаемые в исходное состояние.

Поскольку интегральная схема является неремонтируемым элементом, то в схемах с обратными связями, выполненными в одной ИС, не требуется разрыва связи. При правильных сигнатурах на входе этой схемы и неправильной на выходе (на последнем элементе схемы, с которого снимается обратная связь) эта ИС подлежит замене.

Если же схема с обратной связью выполнена на нескольких интегральных схемах и не предусмотрена ее установка в исходное состояние, то необходимо предусмотреть возможность разрыва обратной связи.

Для схем с обратными связями, устанавливаемых в исходное состояние, введение дополнительного оборудования не обязательно, поскольку их проверку с помощью сигнатурного анализатора можно произвести в однократном режиме.

Применение же индивидуальных внутрисхемных тестеров делает ненужным введение дополнительного оборудования для разрыва обратных связей.

Поиск неисправности в микропроцессорном устройстве может производиться как в автономном режиме, когда стимуляция схем проверяемого устройства осуществляется при помощи аппаратуры самого проверяемого устройства, так и под управлением тестовых последовательностей сигнатурного анализа. Общий алгоритм поиска неисправности в микропроцессорном устройстве представлен на рис. 4.16. Поиск неисправности в микропроцессорном устройстве, которое может быть как активным (содержащим микропроцессоры), так и пассивным (не содержащим микропроцессоров), производится с предварительной проверкой при помощи функциональных тестов или только с помощью сигнатурного анализа, если предварительная частичная диагностика невозможна.

Предварительная проверка устройства с помощью функциональных тестов предназначена для сокращения времени поиска неисправности и дает существенное сокращение этого времени из-за большого количества ручных операций при поиске неисправности с использованием сигнатурного анализа. Выявленная при предварительном тестировании

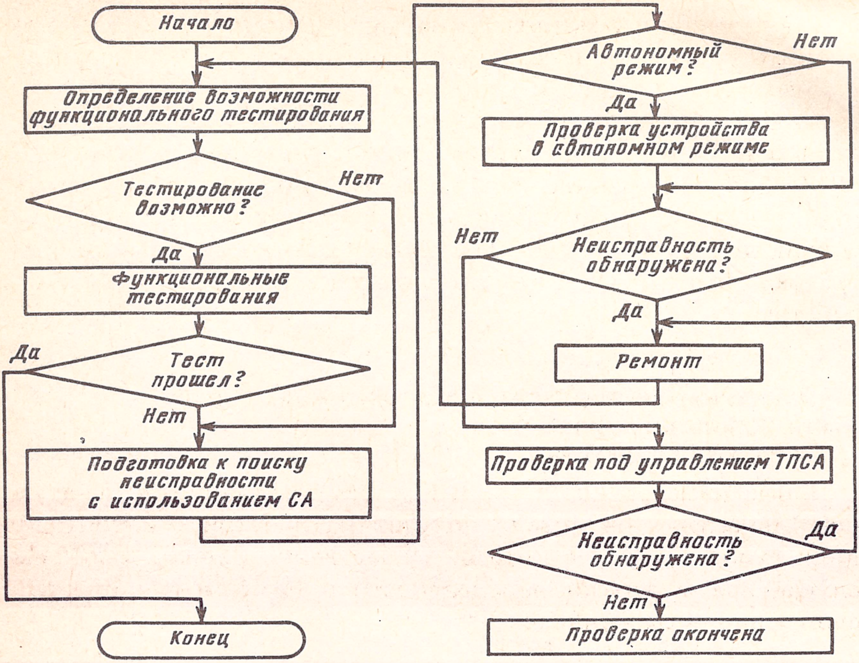


Рис. 4.16. Обобщенный алгоритм поиска неисправности в микропроцессорном устройстве

группа схем затем проверяется с помощью сигнатурного анализа, с помощью которого и локализуется неисправность.

В тех случаях, когда предварительная проверка устройства с помощью функциональных тестов невозможна, поиск неисправности производится только с помощью сигнатурного анализа. Поиск неисправности производится в двух режимах: автономном с аппаратной стимуляцией или (и) под управлением ТПСА. При поиске неисправности используется метод раскрутки. После обнаружения неисправности устройство ремонтируется и повторно проверяется.

Конкретный алгоритм для проверки конкретного устройства определяется исходя из общего алгоритма в зависимости от схемы отказавшего устройства и характера неисправности.

При разработке дополнительного оборудования, необходимого для организации системы диагностирования МПК

с использованием метода сигнатурного анализа, необходимо оценить возможность сохранения связи между МПК, периферийным оборудованием и объектами управления с точки зрения допустимости сохранения связи при поиске неисправности и обеспечения стимуляции проверяемых устройств МПК. Если сохранение связи невозможно или при ее сохранении нельзя стимулировать устройства ввода-вывода или устройства связи с объектом, то необходимо произвести разработку имитатора, который позволит проверить эти устройства.

4.4. СОКРАЩЕНИЕ ВРЕМЕНИ ПОИСКА НЕИСПРАВНОСТИ ПРИ ИСПОЛЬЗОВАНИИ МЕТОДА СИГНАТУРНОГО АНАЛИЗА

Поиск неисправности в МПК с использованием сигнатурного анализа является полуавтоматической процедурой, доля ручного труда в которой существенно велика. В общем случае время поиска неисправности в МПК можно оценить следующим выражением:

$$T = \sum_{i=1}^{m} t_{1i} + \sum_{j=1}^{f} t_{2i} l_j + t_{2f+1} k, \qquad (4.9)$$

где T – время поиска неисправности; t_{1i} – время, необходимое на i-ю коммутацию управляющих сигналов сигнатурного анализатора; t_{2i} – время оценки оператором сигнатуры на одном выводе ИС для i-й коммутации; i – проверка, для которой выполнены коммутации; m – суммарное количество коммутаций управляющих сигналов при проверке МПК; l_j – количество выводов ИС, сигнатуры которых проверяются при j-й коммутации; f – количество коммутаций, в которых неисправность не была обнаружена; k – количество выводов ИС, синатуры которых были проверены до локализации неисправности для коммутации, в которой она была обнаружена.

Как показывает практическое использование сигнатурного анализатора, время поиска неисправности в основном определяется вторым и третьим слагаемыми в (4.9). Так,

например, при поиске неиспрвности в центральном процессорном устройстве [84] в режиме свободного пересчета адресов проверяется приблизительно 180 различных контактов при одинаковой коммутации управляющих сигналов сигнатурного анализатора. Поэтому переход к поиску неисправности в МПК с использованием сигнатурного анализатора по контрольным точкам дает значительный выигрыш во времени.

Поскольку поиск неисправности с использованием сигнатурного анализа существенно отличается от традиционных программных или микропрограммных методов диагностики, то необходимо определить требования к контрольным точкам при таком поиске:

контрольная точка должна быть доступна для контактного приспособления сигнатурного анализатора;

контрольная точка должна по возможнсоти быть выходным контактом схемы, который является логическим концом некоторой части схемы;

в контрольной точке должна отразиться любая неисправность, возникшая в данной части схемы;

множество контрольных точек устройства должно покрывать всю его схему так, чтобы любая неисправность, возникшая в устройстве, отразилась бы в одной из контрольных точек, т.е. изменила сигнатуру хотя бы в одной из контрольных точек;

количество контрольных точек должно быть минимально.

Одним из основных требований сигнатурного анализа к контрольным точкам является требование их доступности. Так, например, согласно требованию программной доступности один из разрядов регистра общего назначения микропроцессора может являться контрольной точкой, а выход буфера шины данных микропроцессора не может служить контрольной точкой, поскольку он программно недоступен. В случае расстановки контрольных точек для сигнатурного анализа, контрольной точкой может служить выход буфера шины данных, так как именно с него может быть подана информация о поведении разряда регистра общего назначения в сигнатурный анализатор.

Сформулированные требования к контрольным точкам позволяют осуществить выбор контрольных точек с помощью языка и методов теории графов. Построим для проверяемой схемы направленный граф $G(E, U)$, в котором каждой вер-

шине графа $e_i \in E$ поставим в соответствие i-й выходной контакт, а дуге $u_{ij} \in U$ – связь между i-м выходным и j-м входным контактами.

Таким образом, задачу выделения контрольных точек можно разделить на несколько этапов, которые на языке теории графов могут быть сформулированы следующим образом:

1) определение множества E_1 висячих вершин графа $G(E, U)$;

2) пересчет путей, связанных с полученным множеством висячих вершин;

3) определение множества E_3 вершин, недоступных с висячих вершин;

4) выделение из полученного в п. 3 множества вершин E_3 минимального множества вершин E_4, которые связаны путями произвольной длины со всеми вершинами множества E_3;

5) объединение множества висячих вершин E_1 и множества вершин, полученных в результате выполнения п. 4, в множество контрольных точек E_k.

Такая постановка задачи позволяет составить формализованную методику выделения контрольных точек:

1. Для решения задачи определения множества висячих вершин E_1 графа $G(E, \Gamma)$ можно воспользоваться следующим определением: вершина e_i висячая, если $\Gamma e_i = \phi$ [77]. Граф $G(E, \Gamma)$ представляется в виде матрицы смежностей. Висячие вершины в этой матрице имеют строки, равные нулю.

2. Пересчет путей, связанных с полученным множеством висячих вершин E_1, можно выполнить, решив задачу нахождения обратного транзитивного замыкания $\hat{\Gamma}^{-} e_i$ для каждой $e_i \in E_1$. В результате решения получаем множество E_2 вершин, связанных с E_1.

3. Из графа $G(E, \Gamma)$ убираются вершины, принадлежащие множествам E_1 и E_2. Оставшиеся вершины представляют собой множество E_3 вершин, не связанных с E_1. В полученном подграфе $G_1(E_3, \Gamma_3)$ определяется минимальное множество E_4 вершин, связанных со всеми вершинами множества E_3.

4. Для всех вершин, являющихся выходами подграфа $G_1(E_3, \Gamma_3)$, определяется обратное транзитивное замыкание $\hat{\Gamma}^{-} e_{3i}$.

Из множества обратных транзитивных замыканий выбирается обратное транзитивное замыкание, имеющее путь максимальной длины. Вершины, вошедшие в это обратное транзитивное замыкание, убираются из подграфа $G_1(E_3, \Gamma_3)$. Полученный подграф $G_2(E_3', \Gamma_3')$ анализируется аналогичным образом. Процедура повторяется до тех пор, пока множество E_n станет пустым.

5. Определяется множество контрольных точек E_3 как объединение множеств E_1 и E_4.

Выделение контрольных точек рассмотрим на примере графа $G(E, \Gamma)$ (рис. 4.17), который представляет собой графовую модель упрощенной схемы магистралей адреса и данных микропроцессорного устройства. В соответствие микропроцессору поставлена вершина A, буферным схемам магистрали адреса соответствует вершина B, вершины C и E соответствуют дешифраторам адреса для дешифрации номера порта ввода-вывода и адреса ПЗУ. Эти дешифраторы вырабатывают сигналы выборки кристаллов для перечисленных схем. Вершине D соответствуют буферные схемы, связывающие магистраль адреса устройства с межмашинной магистралью МПК, вершинам F и G – коммутаторы адреса ОЗУ. Вершины H, L соответствуют в порядке очередности БИС ввода-вывода, корпусу БИС ПЗУ, K и M – БИС динамического ОЗУ. Вершине P – буферные формирователи шин на магистрали данных.

Для $G(E, \Gamma)$ составляется матрица (рис. 4.18), каждая строка которой анализируется на 0, в результате анализа получим

$$\Gamma D = 0. \tag{4.10}$$

Следовательно, вершина D является висячей и должна быть контрольной точкой. А множество E_1 равно

$$E = \{ D \} . \tag{4.11}$$

На поиск обратного транзитивного замыкания наложим ограничение его длины вершиной A, поскольку вход вершины A относится к нерассматривемой в данном случае шине данных микропроцессора. Определим $\hat{\Gamma}^{-}D$, анализируя столбцы матрицы, начиная со столбца D:

$$\hat{\Gamma}^{-}D = \{ D, B, A \} . \tag{4.12}$$

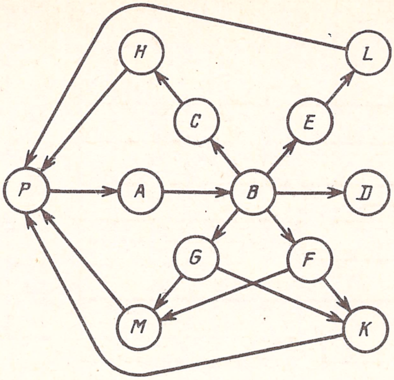


Рис. 4.17. Граф $G(E, Г)$

	A	B	C	D	E	F	G	H	L	K	M	P
A		1						1				
B			1	1	1	1	1					
C								1				
D												
E									1			
F										1	1	
G										1	1	
H												1
L												1
K												1
M												1
P	1											

Рис. 4.18. Матрица смежностей графа $G(E, Г)$

Из графа $G(E, Г)$ убираются вершины, принадлежащие множествам E_1 и E_2. Полученный в результате подграф G_1 (E_3, $Г_3$) представляет собой подграф для множества $Г_3$ вершин, недоступных с висячих вершин.

Для рассматриваемого примера подграф G_1 (E_3, $Г_3$) представлен на рис. 4.19, матрица – на рис. 4.20. На поиск обратных транзитивных замыканий накладывается ограничение: не рассматриваются связи с вершиной P, поскольку она не относится к рассматриваемой схеме. В данном случае обратные транзитивные замыкания равны нулю. Следовательно, все вершины, являющиеся выходами и не попадающие под наложенные ограничения, являются контрольными точками:

$$E_4 = \{ C, E, F, G \} . \tag{4.13}$$

Последним этапом определения множества контрольных точек E является объединение множеств E_1 и E_4.

Для рассмотренного примера множество контрольных точек получается объединением (4.11) и (4.13):

$$E_k = \{ D \} \cup \{ C, E, F, G \} = \{ C, D, E, F, G \} . \tag{4.14}$$

Таким образом, в качестве контрольных точек в рассмотренной схеме должны служить выходы формирователей шин D, дешифраторов C и E, коммутаторов G и F.

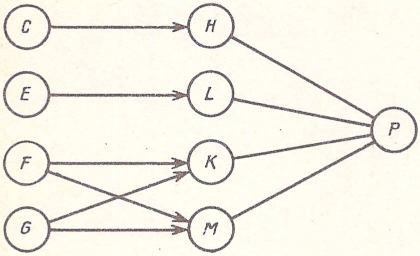


Рис. 4.19. Подграф $G_1(E_3, Г_3)$

	C	E	F	G	H	L	K	M	P
C					1				
E						1			
F							1	1	
G							1	1	
H									1
L									1
K									1
M									1
P									

Рис. 4.20 Матрица смежностей подграфа $G_1(E_3, Г_3)$

Еще более существенного ускорения поиска неисправности можно добиться, применяя метод характеристических сигнатур [85]. Характеристической сигнатурой (ХС) "Окна измерения" именуется сигнатура сигнала Лог. 1, полученная в данном "Окне измерения". Для использования ХС как идентификатора правильного выполнения тестовой последовательности сигнатурного анализа в тестовую последовательность вводится анализ реакции проверяемой схемы на тестовое воздействие с выходом программы на гашение "Окна измерения" в случае ошибки. При этом способе проверки, когда в МПК невозможен запуск функциональных тестов, метод характеристических сигнатур позволяет, используя алгоритм раскрутки, существенно сократить время поиска неисправного устройства. Дальнейшая же локализация отказа производится по основному алгоритму поиска неисправности в устройстве с использованием сигнатурного анализа. При этом формула (4.19) преобразуется в следующую:

$$T = \sum_{i=1}^{m} t_{1i} + \sum_{i=1}^{f} t_{2i} l_i + t_{2p} p + t_{2f+1} k, \qquad (4.15)$$

где p – количество устройств, проверяемых методом характеристических сигнатур.

Основным документом сигнатурного анализа является словарь эталонных сигнатур. Этот документ определяет алгоритм поиска неисправности.

Таблица 4.2

№ оператора	Время, затрачиваемое на проверку 20 точек		№ оператора	Время, затрачиваемое на проверку 20 точек	
	подряд	вразброс		подряд	вразброс
1	2 мин 10 с	4 мин	6	1 мин 50 с	4 мин
2	1 мин 30 с	3 мин 50 с	7	2 мин	4 мин 15 с
3	3 мин 15 с	5 мин 55 с	8	2 мин 45 с	4 мин 15 с
4	2 мин 20 с	4 мин 40 с	9	2 мин 05 с	4 мин
5	2 мин 12 с	4 мин 30 с	10	3 мин	5 мин 55 с

Алгоритм проверки задается либо в порядке перечисления номеров ИС в таблице, либо специальным выделением в таблице контактов ИС, служащих контрольными точками. Как уже было показано выше, поиск неисправности по контрольным точкам значительно ускоряет процесс локализации неисправности. Существенным образом также влияет на скорость проверки и организация словаря эталонных сигнатур. Обычно словарь эталонных сигнатур строится по цепям в прямом или обратном порядке (от контрольной точки или к контрольной точке). При такой организации контрольные точки бывают разбросаны по различным листам словаря эталонных сигнатур.

Однако, как показала практика, организация словаря эталонных сигнатур, при которой все контрольные точки данной проверки располагаются подряд, а при обнаружении неправильной сигнатуры оператор отсылается к проверке цепи, позволяет почти в два раза сократить время проверки устройства.

Проверка производилась следующим образом: десяти операторам было дано задание проверить по 10 раз 20 точек подряд и вразброс. Результаты предствлены в табл. 4.2. Для каждого оператора приведены последние, десятые результаты по каждой проверке. Среднеарифметическое время проверки подряд 20 точек составляет

$$t_{ср1} = \frac{\sum_{i=1}^{10} t_{1i}}{10} = 2\text{ мин }31\text{ с}. \tag{4.16}$$

Среднеарифметическое время проверки 20 точек вразброс составляет

$$T_{ср2} = \frac{\sum_{j=1}^{10} t_{2j}}{10} = 4 \text{ мин } 52 \text{ с.} \qquad (4.17)$$

Сравнение (4.16) и (4.17) показывает, что расположение контрольных точек подряд позволяет сократить время поиска неисправности приблизительно в 1,95 раза.

Глава 5

МЕТОДЫ ПОСТРОЕНИЯ СИСТЕМ ТЕСТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ СИГНАТУРНОГО АНАЛИЗА

5.1. СПОСОБ ФОРМИРОВАНИЯ ТЕСТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ СИГНАТУРНОГО АНАЛИЗА ДЛЯ СТИМУЛЯЦИИ ОТДЕЛЬНЫХ ЧАСТЕЙ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ

Устройства, входящие в состав МПК, в общем случае представляют собой объединение различных схем, реализующих отдельные самостоятельные функции. Для их проверки требуются различные ТПСА, учитывающие структуру, назначение и состав конкретной схемы. Например, пусть некоторое устройство состоит из схем постоянной и оперативной памяти, контроллера параллельного интерфейса, контроллера последовательного интерфейса, тогда для стимуляции этого устройства требуются четыре отдельные тестовые последовательности сигнатурного анализа, созданные для каждого типа схем.

Достоверность функционирования всей системы диагностирования будет определяться не только качеством устройств сигнатурного анализа, но и качеством используемых ТПСА, при этом

$D = D_{Т} D_{А}$,

где D – достоверность системы диагностирования; $D_{Т}$ – достоверность ТПСА; $D_{А}$ – достоверность сигнатурного анализатора.

Здесь необходимо учитывать трудность использования генераторов псевдослучайных чисел в качестве устройств, стимулирующих микропроцессоры, поскольку для микропроцессоров существует некоторое множество запрещенных состояний, а также при такой стимуляции возможна их неполная проверка; это приводит к необходимости применения программой стимуляции.

Тестовые последовательности сигнатурного анализа представляют собой программы, предназначенные для стимуляции проверяемых схем. Они не осуществляют проверки результата воздействия на проверяемую схему, анализ воздействия осуществляется оператором с помощью СА. ТПСА должны удовлетворять следующим требованиям [25, 50, 86, 88–90]:

1) инициализировать сигналы, включающие и выключающие ОИ (сигналы "Пуск" и "Стоп" для СА);

2) стимулировать работу проверяемых схем таким образом, чтобы внутри ОИ они максимально возможное число раз поменяли свое логическое состояние;

3) работать в циклическом режиме;

4) формировать ОИ постоянной длительности, не изменяющиеся от цикла к циклу, т.е. не содержать внутри ОИ условных переходов, изменяющих его длину. Любое изменение ОИ приведет к появлению нестабильных сигнатур и сделает поиск неисправности с помощью СА невозможным;

5) длительность цикла не должна превышать 1,5–2 с, поскольку ОИ с большой длительностью существенно осложняет работу оператора.

Эти требовнаия затрудняют применение функциональных тестов в качестве ТПСА. Хотя принципиально возможно построение функциональных тестов с учетом требований к ТПСА и их использование в этом качестве. Для этого кроме инициализации ОИ в функциональные тесты необходимо заложить свойство, устраняющее все условные переходы внутри ОИ.

Для генерации ТПСА микропроцессора может быть предложен формализованный метод, учитывающий вышеизложенные требования и позволяющий формировать ТПСА

микропроцессора как для полного набора команд, так и для его части. Предлагаемый метод не требует знания топологии и электрической принципиальной схемы микропроцессора.

Рассмотрим микропроцессор в виде композиции двух автоматов – управляющего A и операционного B. Для формального описания микропроцессора будем использовать две алгебры – алгебру операторов $U_о$ и алгебру условий $U_у$ [14]. Моделью микропроцессора служит система объектов $\langle R, V, U \rangle$, где R – множество регистров микропроцессора; V – множество команд; U – множество условий. В [42] доказано, что система $U_а = \langle U_о, U_у \rangle$ является многоосновной алгеброй.

Сформулируем условия, которым должна отвечать система $U_а$, предназначенная для генерации ТПСА микропроцессора, и ряд ограничений, накладываемых на эту систему, а именно:

для разработчика ТПСА известны только система команд и функциональная блок-схема микропроцессора;

ограниченный объем ПЗУ, выделяемый для хранения ТПСА;

необходимость установки в начале и конце ТПСА специальных команд для формирования сигналов, используемых в качестве управляющих для СА;

учет однородности микропроцессора;

высокая повторяемость при формировании операндов;

учет информационной связности операторов между собой таким образом, чтобы в тестовой последовательности образовывалась непрерывная цепочка команд V_i, передающая и преобразующая введеные операнды;

обеспечение подачи операнда на входы микропроцессора и трансляции его после преобразования на выходы прибора;

обеспечение постепенного наращивания проверяемого оборудования от ядра, которое проверяется в режиме аппаратной стимуляции;

каждый регистр микропроцессора проверяется один раз;

арифметические и логические операции проверяются выполнением одной команды из группы однотипных команд.

Маршрутом будем именовать непрерывную цепочку команд V_i, отвечающую требованиям информационной связности операторов и обеспечивающую подачу операторов на входы

микропроцессора в начале цепочки и вывод результата на его выходы в ее конце.

С учетом вышесказанного ТПСА можно именовать набор маршрутов, который обеспечивает полную или частичную проверку всех команд V_i, регистров R_j и множества условий U_k, имеет в своем начале и конце специально вырабатываемые команды для установки сигналов, служащих для включения и выключения ОИ в СА.

Рассмотрим квадратную матрицу $[V_n]$ (табл. 5.1), клетки которой заполнены по следующему алгоритму: единица в клетке V_{ij} устанавливается в том случае, если действие V_j может быть выполнено после действия V_i. Связанный перечень элементов матрицы получим путем выписывания элементов матрицы в порядке, соответствующем расстановке единиц, т.е. $V_k(V_i)$, $V_i(V_j)$, $V_j \ldots$, где в скобках даны индексы элементов, которые соответствуют столбцам матрицы $[V_n]$, образующих клетку V_{ij}. Поставим в соответствие каждому элементу матрицы $[V_n]$ команду из набора команд микропроцессора. При этом команды, осуществляющие ввод информации в микропроцессор (чтение, ввод), в табл. 5.1 имеют не заполненные единицами столбцы. Это означает, что они не зависят от выполнения предыдущей команды, а команды, осуществляющие вывод информации из микропроцессора (запись, ввод), имеют незаполненные строки, означающие, что после них может быть выполнена любая команда. Установим требование, что выписывание команд начинается с команды, имеющей свободный столбец, и заканчивается командой, имеющей свободную строку. Тогда выписанные по предложенному алгоритму команды будут являться маршрутом, удовлетворяющим всем изложенным выше требованиям. Например, из табл. 5.1 можно выписать следующие маршруты:

$V_1 \quad V_3 \quad V_k \quad V_4 \quad V_{n-1}$

$V_2 \quad V_4 \quad V_3 \quad V_n$

$\vdots \quad \vdots \quad \vdots \quad \vdots$

Таблица 5.1

	V_1	V_2	V_3	V_4	...	V_k	...	V_{n-1}	V_n
V_1			1						
V_2				1					
V_3						1			1
V_4			1					1	
⋮									
V_k									
⋮									
V_{n-1}									
V_n									

На базе проведенных рассуждений может быть сформулирована формализованная методика составления ТПСА микропроцессора:

1) строится квадратная матрица $[V_n]$ из системы команд микропроцессора;

2) проставляются 1 в клетках матрицы в соответствии с системой команд, функциональной блок-схемой микропроцессора и перечисленными требованиями;

3) для первого маршрута выписываются команды, проверенные в автономных режимах, и команды передачи между регистрами;

4) с использованием операций из предыдущего маршрута выписывается следующий маршрут с новыми командами;

5) повторяется п. 4 до тех пор, пока не будут выписаны все команды;

6) в начало и конец множества выписанных маршрутов добавляются команды, позволяющие осуществить формирование ОИ для СА, которые должны быть обязательно проверены в автономном режиме.

Проверка операционного автомата U_0, т.е. регистровой части микропроцессора, а также арифметико-логического устройства существенно зависит от выбранной электрической схемы и топологии кристалла. Для проверки регистровой части могут быть рекомендованы операнды, осущест-

вляющие сдвиг на фоне нулей и сдвиг нуля на фоне единиц, а для проверки арифметико-логического устройства – чередующиеся операнды типа 101 и 010 и операнды, позволяющие вырабатывать перенос в арифметических действиях.

Пример. Рассмотрим процесс составления ТПСА для микропроцессора К580ИК80. Предположим, что на этапе автономной проверки была проверена правильность выполнения следующих 11 команд: MVI A; MOV B, A; STA adr; LDA adr; IN; OUT; JMP; INC; JZ; JNZ; NOP.

Выполним п. 1 методики – составим квадратную матрицу (табл. 5.2) из команд микропроцессора. В нашем примере используются 16 команд, хотя реально количество команд определяется системой команд микропроцессора, а при

Таблица 5.2

	MVI A	LXI H	INR A	INX H	MOV B,A	MOV C,B	MOV A,M	MOV M,A	MOV A,C	ADD B	SUB C	XRA H	OK AB	JZ	JNZ	STA adr	OUT
MVI A			1		1			1								1	1
LXT H				1			1	1									
INR A					1			1						1	1	1	1
INX H							1	1									
MOV B,A						1				1			1				
MOV C,B									1		1						
MOV A,M			1		1			1								1	1
MOV M,A																	
MOV A,C								1								1	1
ADD B								1						1	1	1	1
SUB C								1						1	1	1	1
XRA H								1						1	1	1	1
ORA B								1						1	1	1	1
JZ			1	1	1	1	1	1	1	1	1	1	1	1			
JNZ			1	1	1	1	1	1	1	1	1	1	1	1			
STA adr																	
OVT																	

его тестировании в два этапа – командами, используемыми в последующих за ТПСА микропроцессора тестами.

Проставим единицы в клетках матрицы.

Выпишем первый маршрут: MVI A; MOV B, A; MOV C, B; MOV A, C; STA adr. В выписанный маршрут входят две непроверенные команды.

Выпишем следующий маршрут: MVI A; MOV B, A; ADD B; STA adr.

Продолжим выписывание маршрутов до тех пор, пока не будут использованы все команды.

После окончания процесса выписывания маршрутов и добавления к ним команд OUT с адресом программно устанавливаемого триггера тестовая последовательность сигнатурного анализа (ТПСА) готова.

Для организации ТПСА микропроцессора эффективно использование метода характеристических сигнатур [86], блок-схема такой ТПСА представлена на рис. 5.1.

Как правило, наибольшее количество БИС в составе МПК принадлежит оперативному запоминающему устройству. В МПК в основном применяются БИС ОЗУ динамического типа, имеющие наибольшую емкость. ТПСА для проверки БИС ОЗУ может быть построен по любому из известных алгоритмов проверки ОЗУ. Наибольший эффект дают тесты, построенные с учетом топологии кристалла ОЗУ, которые позволяют выявить ошибки, вызванные взаимным влиянием элементов памяти. Однако, как показано в [91, 92], топология кристалла не является тривиальной и внешняя адресация элементов памяти чаще всего не соответствует внутренней. Топология кристалла является производственным секретом и пользователь, как правило, ее не знает.

Хорошую разрешающую способность имеет алгоритм, известный под наименованием "Тест Марш" [93, 94], позволяющий выявлять ошибки, связанные не только с отказом самого элемента памяти, но и частично с взаимным влиянием элементов памяти, одновременной выборкой двух и более элементов.

Наиболее часто в настоящее время в отечественных вычислительных устройствах применяются ОЗУ типа К565 РУ3А. Эти ОЗУ динамического типа с организацией $1 \times n$ (Кбит). Для организации слова длиной в m разрядов необходимо

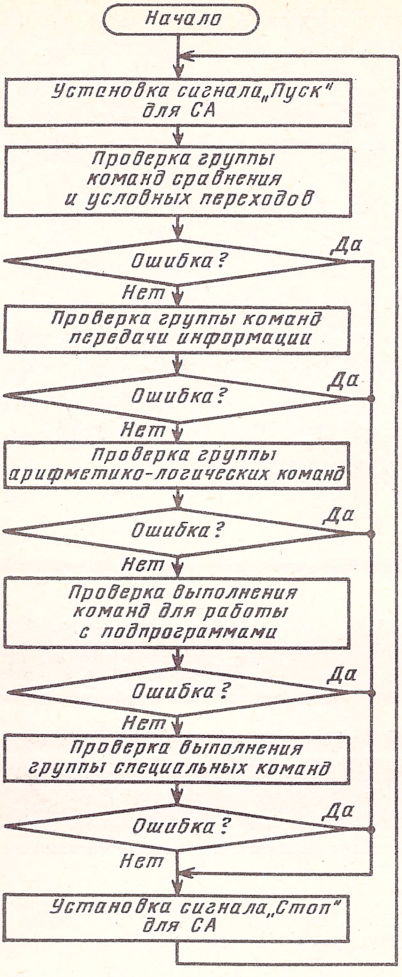


Рис. 5.1. Блок-схема алгоритма ТПСА микропроцессора, построенного по методу характеристических сигнатур

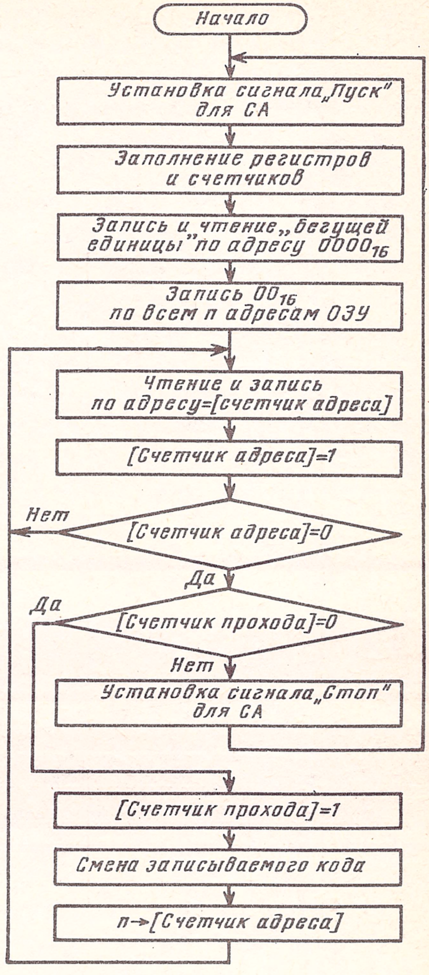


Рис. 5.2. Блок-схема алгоритма тестирования ОЗУ

установить m кристаллов такой памяти. Полученная емкость памяти будет в этом случае равна $m \times n$ Кбит.

Анализ схемы позволяет сделать вывод, что проверка ошибок, связанных с короткими замыканиями между разрядами внутри каждого из n слов, не нужна. Алгоритм тестирования ОЗУ, учитывающий одноразрядную организацию памяти, представлен на рис. 5.2. В этом алгоритме предусмот-

рена проверка, осуществляющая установку каждого разряда в единичное значение при нулевых значениях во всех остальных разрядах слова, однако эта проверка осуществляется только для одного слова из n и предназначена для выявления коротких замыканий в информационных цепях печатной платы.

Константы, применяемые в этом тесте, могут быть произвольными с обязательным условием, чтобы каждый разряд памяти устанавливался в нулевое и единичное значения. Если же в устройстве сделана схема контроля, то эти константы должны учитывать проверку работоспособности кристаллов памяти, хранящих контрольные разряды, Например, для контроля по четности в качестве констант должны быть применены числа, имеющие четное и нечетное количества единиц. Время, затрачиваемое на работу данного алгоритма, определяется по формуле

$$T = 2nt_1 + 2nt_2 + mt_3,$$

где T – время прохождения теста; n – объем кристалла памяти; t_1 – длительность цикла записи в одну ячейку памяти; t_2 – длительность цикла чтения из одной ячейки памяти; m – длина слова памяти; t_3 – длительность цикла записи-чтения при проверке "бегущая 1".

Для использования этого алгоритма в качестве ТПСА ОЗУ в него введена установка сигналов, являющихся сигналами "Пуск" и "Стоп" для формирования ОИ, а также исключен анализ результата.

В данной работе рассматривается проверка только цифровых устройств, устройства аналогового ввода-вывода также могут быть проверены с помощью метода сигнатурного анализа, способ их проверки описан в [79].

5.2. МЕТОДИКА СОСТАВЛЕНИЯ СИСТЕМ ТЕСТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДЛЯ ПРОВЕРКИ ОТДЕЛЬНЫХ УСТРОЙСТВ

Система ТПСА для проверки устройства является упорядоченным множеством тестовых последовательностей, предназначенных для стимуляции отдельных схем, из которых оно состоит. Упорядочение множества ТПСА состоит

в их расстановке согласно алгоритму проверки устройства, т.е. определения последовательности их запуска, в объединении нескольких последовательностей в общем цикле, организации единой системы "Окон измерения" внутри каждого цикла [25, 41, 50, 70, 95].

Создание системы ТПСА для проверки устройства в общем случае осуществляется в два этапа. На первом этапе на предприятии, которое разрабатывает устройства, создается система ТПСА для их базовой конфигурации. На втором этапе при создании системы диагностирования МПК система ТПСА устройства пересматривается, из нее, если это необходимо, удаляются ТПСА неиспользуемых в новой конфигурации схем и добавляются ТПСА, обеспечивающие проверку режимов и схем, не проверяемых в базовой конфигурации.

Создание системы ТПСА устройства на этапе его разработки предлагается производить по методике, изображенной на рис. 5.3. При создании системы производится упорядочение ТПСА устройств с тем, чтобы образованная система и ее части отвечали следующим требованиям:

1. Система ТПСА устройства должна по возможности размещаться в памяти единой областью и вводиться с внешнего носителя за один ввод.

2. Система ТПСА должна обеспечивать простой способ запуска каждой из своих частей.

3. Система ТПСА должна обеспечивать формирование "Окон измерения" при минимальном количестве коммутаций, производимых оператором.

4. В системе образуются группы из ТПСА, объединяемых в одну циклическую последовательность.

Для упрощения запуска системы желательно организовывать единый цикл для системы ТПСА. Простота запуска системы ТПСА и отдельных ее частей важна, когда конфигурация, состав МПК или возникшая неисправность не позволяют запустить программу с пульта управления или дисплея. Однако создание единого цикла в системе ТПСА часто бывает невозможно, поскольку нарушается требование максимальной длительности цикла и количество сигналов, которое можно использовать для формирования ОИ, оказывается недостаточным для создания непрерывной последовательности ОИ.

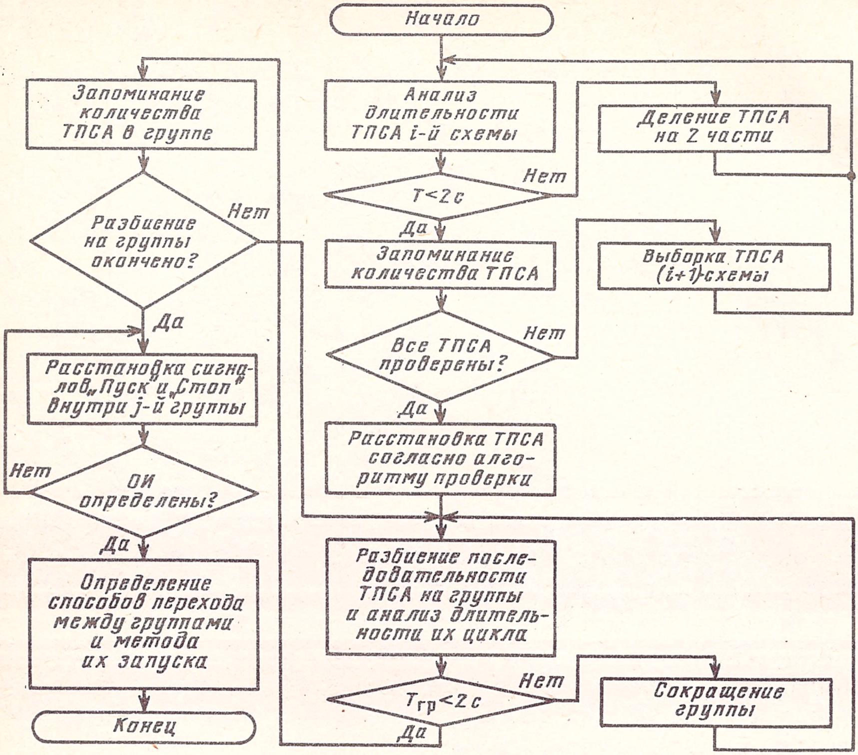


Рис. 5.3. Методика организации системы ТПСА устройства

Условимся именовать группой ТПСА набор тестовых последовательностей, предназначенных для стимуляции отдельных частей устройства, объединенных в единый цикл, имеющий одну точку запуска. Количество ТПСА в группе изменяется от 1 по n, где n – количество ТПСА для полной проверки устройства. При этом если длительность одной ТПСА больше двух секунд, то такая ТПСА разбивается на две или больше частей, каждая из которых становится самостоятельной группой. Общая структура группы ТПСА изображена на рис. 5.4.

Объединение всех групп в единую систему дает систему ТПСА устройства, структура которой отображена на рис. 5.5.

Определение последовательности объединения групп, а также порядок расстановки ТПСА внутри группы опреде-

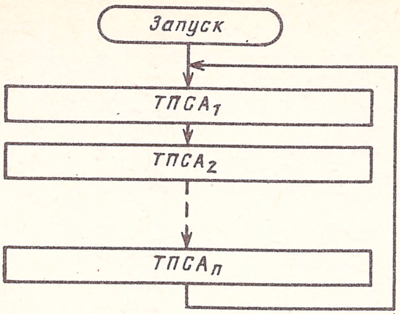


Рис. 5.4. Структура группы ТПСА

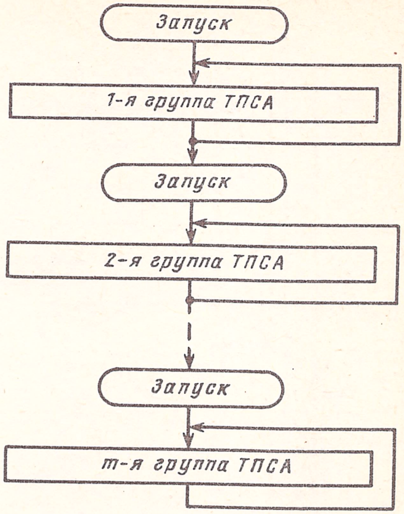


Рис. 5.5. Структура системы ТПСА

ляются алгоритмом проверки, для определения которого может использоваться формализованная методика (см. § 4.2).

Для использования этой методики зададим устройство в виде графа $G(E, V)$, который задан матрицей смежности. Множество вершин $e_i \in E$ представляет собой множество частей устройства, для которых создаются отдельные ТПСА. Проставляется единица в клетке, образованной пересечением i-й строки и j-го столбца, если i-я часть устройства используется для проверки j-й. Полученная матрица смежности анализируется по методике, изложенной в § 4.2. Полученное в результате анализа упорядоченное множество $e_i \in E$ определяет алгоритм проверки устройства и последовательность расстановки ТПСА в системе ТПСА устройства.

Дальнейшая процедура создания системы производится по методике, представленной на рис. 5.3.

Другим важным вопросом является выбор сигналов, используемых в качестве сигналов "Пуск" и "Стоп" для СА, которые должны отвечать требованиям, изложенным в [25, 50, 87, 89, 95]. Количество таких сигналов должно быть минимально. Определение этого количества можно произвести, учитывая возможность гибкого использования в СА различных перепадов одного и того же сигнала в качестве сигналов "Пуск" и "Стоп". Произведем анализ различных способов формирования ОИ (рис. 5.6). Первый способ форми-

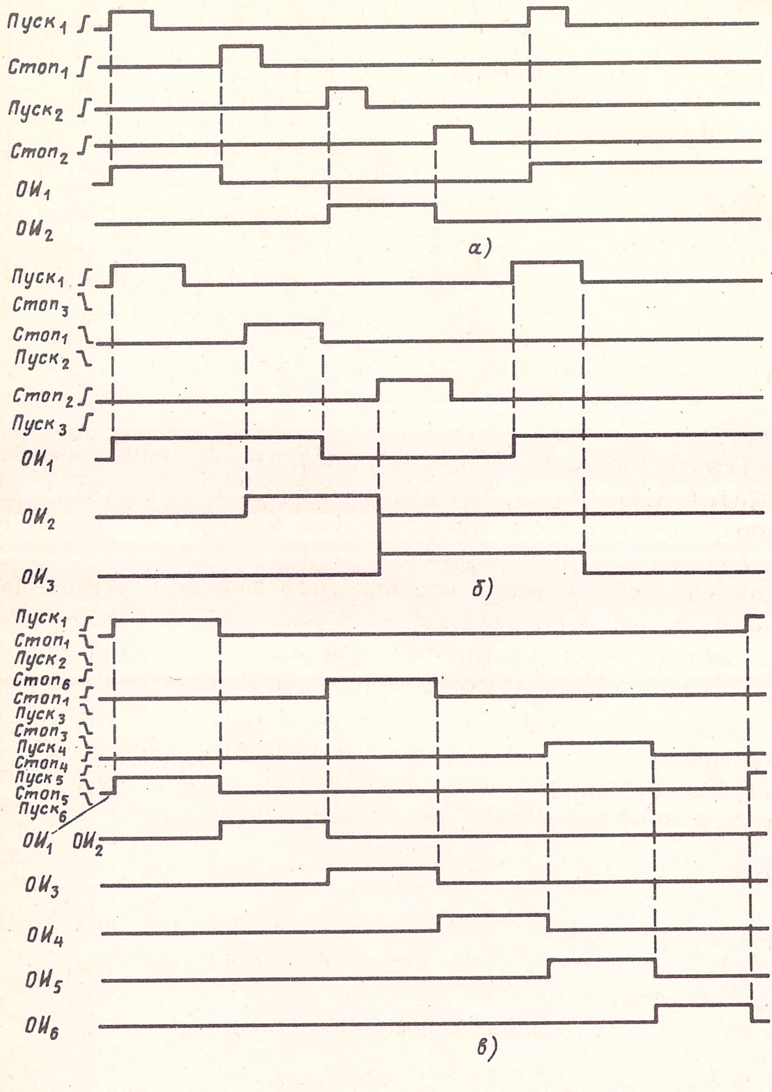


Рис. 5.6. Формирование ОИ с использованием различных (*а*) и совмещенных (*б*) сигналов и с использованием одного и того же сигнала (*в*) в качестве "Пуск" и "Стоп" для нескольких ОИ

рования (рис. 5.6, *а*) требует для организации сигналов "Пуск" и "Стоп" $2N$ сигналов, для организации N ОИ, где N – количество ТПСА в группе. Второй способ организации (рис. 5.6, *б*) ОИ требует N сигналов для N ОИ. Способ, изображенный на рис. 5.6, *в*, позволяет с помощью $N/2$ сигналов, используемых в качестве "Пуск" и "Стоп" для СА, организовать N ОИ.

Способ формирования ОИ и стробирование приема данных позволяют создать формализованную модель устройства для расчета эталонных сигнатур. Большинство микропроцессоров выполняет команды как последовательность машинных циклов, каждый из которых представляет собой набор микрокоманд, предназначенных для выполнения какой-нибудь операции, например: чтения или вывода и т.д. Каждый элемент микропроцессорного устройства или МПК находится в строго определенном состоянии в каждый фиксированный (стробирующим сигналом СИ) момент времени. (Элементы схемы, находящиеся в данный момент времени в произвольном состоянии на данном рабочем перепаде СИ, не проверяются и, следовательно, в данной модели не рассматриваются.)

При создании модели формируется таблица для каждого из машинных циклов, в которой отображается состояние всех элементов устройства для каждого из используемых СИ. Затем для каждой команды формируется сводная таблица, являющаяся объединением таблиц для машинных циклов, входящих в состав данной команды. Сводные таблицы команд для каждого СИ отдельно располагаются сверху вниз в последовательности, определенной расположением команд в ТПСА внутри данного ОИ. В результате в столбцах таблицы получим состояние каждого из элементов устройства во время рабочего перепада СИ внутри ОИ, таким образом, столбец полученной таблицы определяет входную двоичную последовательность для СА на выходе данного элемента устройства.

5.3. ОРГАНИЗАЦИЯ СИСТЕМ ТЕСТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ СИГНАТУРНОГО АНАЛИЗА ДЛЯ МПК

Организация системы тестовых последовательностей сигнатурного анализа для поиска неисправности в МПК является составной частью задачи создания системы диагностирования МПК с использованием сигнатурного анализа.

Эта задача решается предприятием, разрабатывающим данный микропроцессорный комплекс. Предприятие, разрабатывающее МПК, приобретает совместно с набором устройств системы поиска неисправностей на основе метода сигнатурного анализа, которые включают в себя системы ТПСА, предназначенные для стимуляции схем, входящих в состав этих устройств. Таким образом, создание систем ТПСА осуществляется на основе систем ТПСА для поиска неисправностей в устройствах, которые созданы в организации, разработавшей набор микропроцессорных устройств.

Система тестовых последовательностей сигнатурного анализа для проверки МПК может быть определена как упорядоченная совокупность систем тестовых последовательностей сигнатурного анализа для проверки устройств, входящих в его состав. При создании системы ТПСА МПК необходимо:

1) установить расположение программ ТПСА устройств в порядке, определенном алгоритмом проверки МПК;

2) определить тип носителя для хранения ТПСА, способ их загрузки в ОЗУ МПК;

3) создать программы, обеспечивающие связь между отдельными системами ТПСА устройств, их последовательную загрузку и запуск;

4) осуществить распределение массива памяти в МПК для хранения ТПСА в процессе проверки.

Методика создания системы ТПСА МПК представлена на рис. 5.7. Следует пояснить ряд пунктов данной методики.

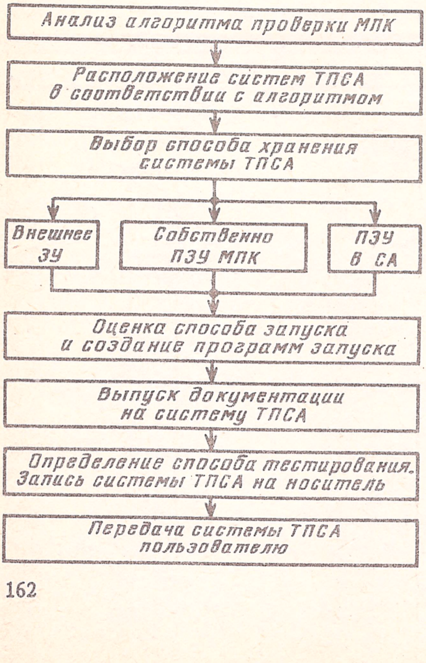


Рис. 5.7. Методика создания системы ТПСА для проверки МПК

Выбор способа хранения системы ТПСА определяется составом МПК и типом СА. Наиболее простым является использование для хранения ТПСА памяти, расположенной в СА. В этом случае ПЗУ МПК может вообще не использоваться для целей хранения диагностических программ. Все остальные способы хранения ТПСА зависят от наличия в составе МПК того или иного ВЗУ и характера отказа. При отсутствии ВЗУ или если отказ не позволяет осуществить ввод информации с ВЗУ, то ТПСА приходится хранить в ПЗУ МПК, уменьшая при этом объем рабочих программ. Альтернативой этому является способ, при котором кристаллы БИС ПЗУ с ТПСА устанавливаются в разъемы ПЗУ вместо БИС с рабочими программами, что, однако, приводит к уменьшению срока службы этих разъемов.

Выбор способа запуска ТПСА зависит от состава МПК и типа отказа. Запуск любой программы, в том числе и ТПСА, производится с пульта управления или дисплея, однако характерной особенностью МПК, работающих в производственных условиях, является отсутствие этих устройств и программ-мониторов, обеспечивающих обмен информацией с дисплеями, пультами управления и т.п. В этих условиях необходимо пользоваться той особенностью микропроцессорной техники, которая заключается в установке по сигналу "Сброс" определенного фиксированного адреса, определяющего начало выполняемой программы. Поэтому первая БИС с ТПСА должна устанавливаться по этому адресу. Эти БИС должны содержать программу связи ТПСА, обеспечивающую их поочередный запуск. Для перехода от одной системы ТПСА к другой могут быть использованы сигналы прерывания, искусственно возбуждаемые на входах контроллера прерывания, или другие способы аппаратной установки признака перехода к следующей ТПСА. Более простым способом является последовательная физическая замена одних БИС с ТПСА на другие.

Необходимо выделить в ОЗУ МПК две области памяти, одна из которых используется в качестве рабочей зоны, а другая – для хранения выполняемых ТПСА. Поскольку адресация памяти при создании МПК может изменяться (по сравнению с базовой конфигурацией), то нужно произвести переадресацию ТПСА и изменить адреса ячеек, используемых в качестве рабочих. При выделении областей памяти для

хранения тестовых последовательностей как в ПЗУ, так и в ОЗУ необходимо учесть, что эти области памяти должны быть проверены до их использования.

Для системы ТПСА МПК должна быть выпущена документация, в которую должны войти блок-схема ТПСА МПК и блок-схема ТПСА устройств, расположенных в порядке, определенном блок-схемой ТПСА комплекса, тексты всех программ, входящих в систему, а также руководство оператора, в котором должно быть простым, доступным языком описаны правила их запуска.

Глава 6
РАЗРАБОТКА СИСТЕМЫ ДИАГНОСТИРОВАНИЯ МИКРОПРОЦЕССОРНОГО КОМПЛЕКСА С ИСПОЛЬЗОВАНИЕМ СИГНАТУРНОГО АНАЛИЗА

6.1. УСТРОЙСТВО СИГНАТУРНОГО АНАЛИЗА ДЛЯ ПОИСКА НЕИСПРАВНОСТЕЙ В МПК, РЕАЛИЗОВАННОМ НА БАЗЕ МСУВТ В7

Разработка устройства диагностики СЛ01П производилась с учетом особенностей устройств, входящих в состав МСУВТ В7 [96], поэтому в него введены некоторые дополнительные возможности по сравнению с универсальными сигнатурными анализаторами.

Блок-схема устройства диагностики СЛ01П представлена на рис. 6.1. Основные технические характеристики СЛ01П следующие:

Сигнатура	4-разрядное 16-ричное число
Алфавит сигнатуры	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, C, F, H, P, U
Вероятность распознавания правильной входной последовательности данных	100 %
Вероятность распознавания ошибочной входной последовательности данных	99,998 %
Входной импеданс сигналов, поступающих в СЛ01П через жгут ЭСС25	48 кОм

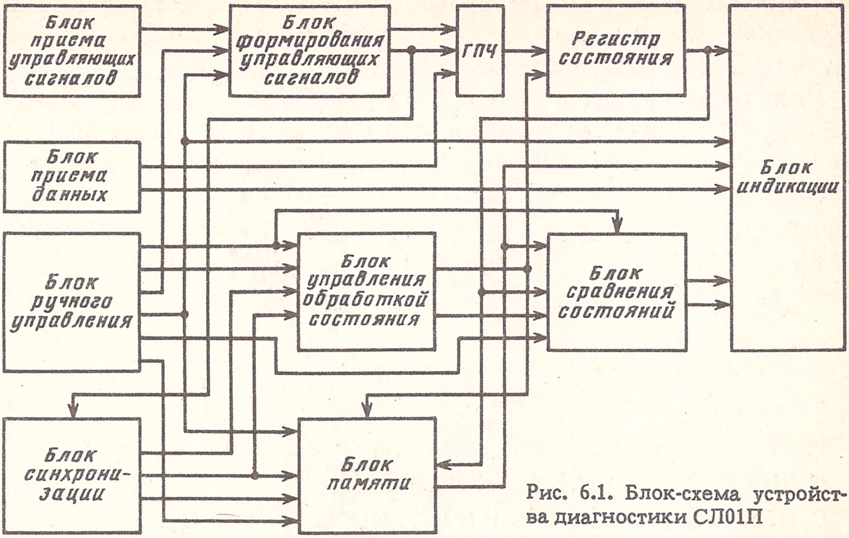


Рис. 6.1. Блок-схема устройства диагностики СЛ01П

Входные управляющие сигналы:	
сигнал, определяющий начало ОИ	"Пуск"
сигнал, определяющий окончание ОИ	"Стоп"
синхронизирующий сигнал .	СИ
дополнительный синхронизирующий сигнал, не принимающий участия в формировании ОИ	ОЖ
Сдвиг начала ОИ относительно сигнала "Пуск" на	2^{16} синхроимпульсов
Сдвиг конца ОИ относительно сигнала "Пуск" на	2^{16} синхроимпульсов
Минимальное время предустановки данных и управляющих сигналов относительно рабочего перепада СИ	25 нс
Минимальное время задержки данных и управляющих сигналов относительно рабочего перепада СИ	0 нс

Устройство диагностики СЛ01П выполнено по второму конструктивному решению, предложенному в § 3.1 (рис. 3.4). При поиске неисправности на место проверяемого печатного узла вставляется удлинитель, имеющий два ряда разъемов – горизонтальный и вертикальный. В горизонтальный ряд разъемов удлинителя вставляется проверяемое устройство, а в вертикальный – СЛ01П. Управляющие сигналы могут подаваться в СЛ01П как через разъемы удлинителя, так и через специальный жгут ЭСС25, который подсоединяется

к СЛ01П при помощи кабельного разъема. Жгут ЭСС25 имеет контактные приспособления – контактодержатели, которые обеспечивают надежное контактирование с выводом ИС или другого электрорадиоэлемента (ЭРЭ) в течение длительного времени без помощи человека. Данные с выводов проверяемого устройства подаются в СЛ01П через контактирующее устройство, имеющее контакт игольчатого типа, которое в дальнейшем именуется пробником. В пробнике кроме контакта игольчатого типа установлена кнопка "Сброс", позволяющая установить схемы СЛ01П в исходное состояние.

Перечисленные выше конструктивные особенности организации системы поиска неисправности в МСУВТ В7 позволили:

использовать для питания СЛ01П источник питания МПК;

осуществить связь между СЛ01П и проверяемым устройством через разъемы удлинителя, что дало возможность существенно упростить процедуру перевода проверяемых устройств в автономный режим при поиске неисправности и передавать управляющие сигналы непосредственно в СЛ01П, без ручной коммутации контактирующих приспособлений жгута ЭСС25;

упростить процедуру поиска неисправности, поскольку оператору не требуется стол или подставка для установки сигнатурного анализатора, что особенно важно при поиске неисправности в МПК, установленных в шкафах.

Как показали опыт эксплуатации СЛ01П и анализ современного состояния разработок сигнатурных анализаторов и других устройств диагностики для микропроцессорной техники, перспективным направлением в области создания СА является создание автоматизированных СА с использованием микропроцессоров, которые должны иметь возможность работы в режиме внутрисхемных эмуляторов. Кроме того, СА должны автоматически обеспечивать алгоритмы проверки, задавая оператору адреса контрольных точек и автоматически сравнивая сигнатуры. Эти функции могут быть реализованы, например, с помощью устройства связи с проверяемым устройством, позволяющим работать в реальном масштабе времени [53].

6.2. РАЗРАБОТКА ЦЕНТРАЛЬНОГО ПРОЦЕССОРНОГО УСТРОЙСТВА С УЧЕТОМ ТРЕБОВАНИЙ ДИАГНОСТИКИ И АЛГОРИТМЫ ПРОВЕРКИ УСТРОЙСТВ МСУВТ В7

Центральное процессорное устройство МСУВТ В7 ПМВ01П разработано с применением микропроцессорного комплекта БИС К580 [84]. В ПМВ01П входят: микропроцессор К580ИК80, ППЗУ 16к×8, ОЗУ 16к×8, программируемый последовательный интерфейс на базе КР580 ВВ51А, параллельный программируемый интерфейс из двух КР580 ВВ55, два интервальных таймера КР580ВИ53, две БИС контроллеров прерывания КР580ВН59.

Кроме перечисленных БИС в ПМВ01П входят различные схемы сопряжения, дешифрации и управления, выполненные на ИС и СИС серий К589, К155, К531.

На рис. 6.2 представлены упрощенная структурная схема ПМВ01П и части межмашинной магистрали с устройством арифметического расширителя ПСВ01П и устройством цифрового ввода-вывода ЦВВ01П. Устройство ПМВ01П имеет магистральную организацию. С целью улучшения контролепригодности и сокращения нагрузки шинных формирователей внутриустройственной магистрали данных подсоединение устройств к ней производится с использованием двух дополнительных магистралей данных: магистрали данных памяти, выполненной на формирователе шин (ФШ) *6*, и магистрали данных ввода-вывода, которая образована с использованием формирователя шин *4*. При этом повысилась надежность внутренней магистрали данных, поскольку в этом варианте организации к ней подсоединяется не 25 различных БИС, а только четыре.

Как показывалось выше, неисправности, искажающие информацию на шине данных внутренней магистрали, приводят к неправильной работе микропроцессора (даже, если он исправен), который будет генерировать неправильные тестовые последовательности. В этом случае будут искажены сигнатуры во многих точках схемы и локализация неисправностей становится невозможной.

Для реализации требований сигнатурного анализа к проверяемому устройству и осуществления поиска неисправности по методу раскрутки, необходимо обеспечить возможность отключения большинства узлов от шины данных

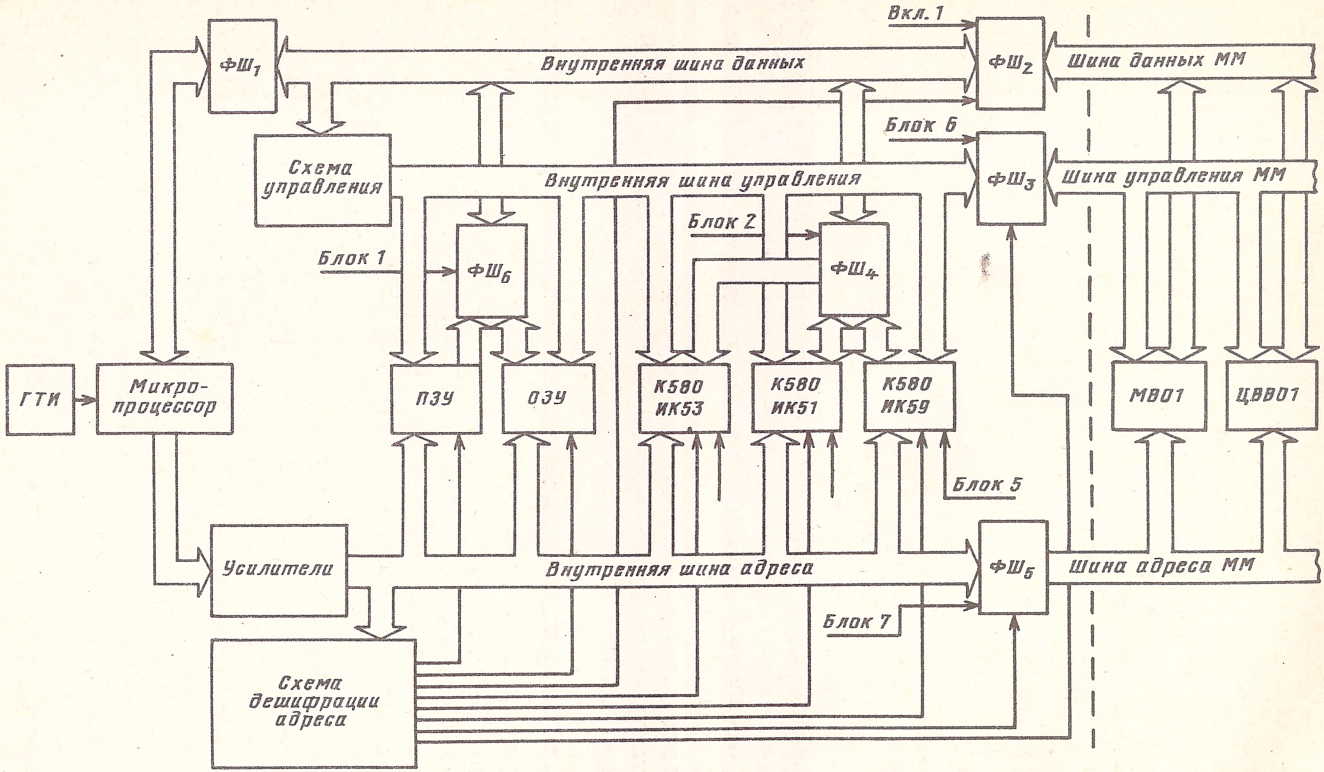


Рис. 6.2. Упрощенная структурная схема ПМВ01П

в начале диагностирования и их последовательное подключение в ходе наращивания проверяемого оборудования. С этой целью введены специальные сигналы блокировки "Блок 1" – "Блок 7", показанные на рис. 6.2, которые позволяют отключать блокируемые устройства от магистрали данных. Сигналы "Блок 1" – "Блок 7" и "Вкл. 1" выведены на разъемы устройства ПМВ01П, стыкующиеся с разъемами блока микропроцессорного комплекса, что позволяет осуществлять блокировку устройств при помощи тумблерного набора, расположенного в устройстве диагностики СЛ01П.

Для разработки алгоритмов поиска неисправности в ПМВ01П с использованием сигнатурного анализа схема микроУВМ ПМВ01П была разделена на три части;

ядро оборудования, работоспособность которого должна быть гарантирована, состоящее из блока питания и генератора тактовых импульсов и схем блокировки;

расширенное ядро оборудования, работоспособность которого может быть проверена в автономном режиме с использованием аппаратной стимуляции, состоящее из микропроцессора и схем трансляции, дешифрации и обработки адреса, а также ПЗУ;

остальное оборудование, проверяемое с использованием стимуляции при помощи ТПСА.

Емкость ПЗУ, выделенная для хранения ТПСА в ПМВ01П, была ограничена одним корпусом БИС ПЗУ, минимальная емкость которого равна 512 байт (в ПМВ01П предусмотрена возможность установки БИС ПЗУ различных типов, из всех возможных БИС для хранения ТПСА был выбран тип ПЗУ, имеющий минимальную емкость памяти). Поэтому та часть оборудования, которая проверяется с использованием программной стимуляции, также была разделена на две группы, в первую из которых вошли схемы, обеспечивающие ввод, хранение и исполнение ТПСА, а во вторую все остальные схемы. Ввод основной части ТПСА предусмотрен с кассетного магнитофона. Исходя из вышесказанного определен алгоритм поиска неисправности в центральном процессорном устройстве ПМВ01П (рис. 6.3).

На первом этапе проверки оператор устанавливает все блокировки и сигнал "Вкл. 1". При этом к микропроцессору подключены внутренняя и внешняя магистрали данных, а все устройства, подключенные к ним, заблокированы.

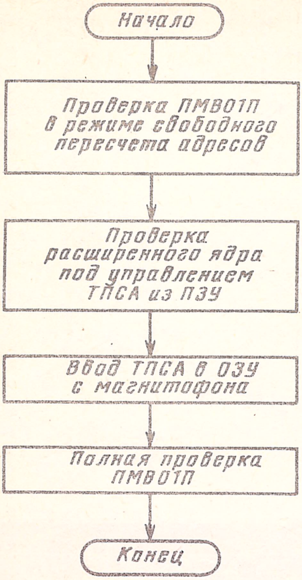


Рис. 6.3. Блок-схема алгоритма поиска неисправности в ПМВ01П

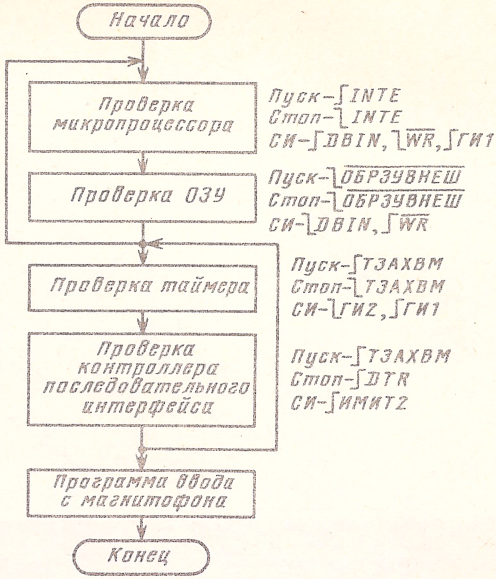


Рис. 6.4. Блок-схема алгоритма поиска неисправности в ПМВ01П под управлением ТПСА из ПЗУ

Шины внешней магистрали данных при отключенных устройствах расширения установлены в состояние Лог. 1. Эти сигналы, пройдя через инвертирующие формирователи шин, устанавливают на шинах данных микропроцессора код команды NOP–00_{16}. Выполняя эту команду, микропроцессор сканирует все поле адреса.

В качестве сигналов "Пуск" и "Стоп" СА используется старший разряд адреса микропроцессора. В этом случае ОИ охватывает все адресное поле. Взяв в качестве СИ синхросигнал "Прием" микропроцессора и выбрав в качестве рабочего перепада спад этого сигнала, оператор проверяет сигнатуры на всех адресных шинах и схемах дешифрации и управления, а также правильность считывания информации из ПЗУ ПМВ01П. Объем проверенного в этом режиме оборудования достаточен для запуска ТПСА из ПЗУ ПМВ01П.

После окончания проверки в режиме аппаратной стимуляции оператор вынимает из разъема корпус БИС ПЗУ с на-

чальным адресом 0000_{16}, вставляет на его место БИС ПЗУ с ТПСА и приступает ко второму этапу проверки, алгоритм которого приведен на рис. 6.4.

Как видно из рис. 6.4, алгоритм поиска неисправности под управлением ТПСА из ПЗУ ПМВ01П представляет собой систему ТПСА, упорядоченную при помощи организации единой системы "Окон измерения", задания строгой последовательности исполнения и определения циклов внутри системы ТПСА. Выбор циклов внутри системы ТПСА определялся длительностью "Окон измерения" внутри цикла. Для данного случая общая длина всех четырех ОИ составляет приблизительно 4 с, что делает неудобным для оператора поиск неисправности, поэтому в данной системе ТПСА было организовано два цикла с общей длительностью приблизительно 2 с.

Под управлением системы ТПСА из ПЗУ ПМВ01П оператор проверяет такие схемы, как микропроцессор, ОЗУ и схемы, связанные с ним, интервальный таймер, контроллер последовательного ввода-вывода, устройство связи с магнитофоном. Все устройства, за исключением ОЗУ, в этой системе ТПСА проверяются частично, что вызвано экономией емкости ПЗУ.

После окончания проверки при отсутствии неисправности (или после ремонта, в случае обнаружения неисправности и проведения последующей повторной проверки) оператор приступает к заключительному этапу полной проверки ПМВ01П под управлением системы ТПСА из ОЗУ ПМВ01П, алгоритм которой представлен на рис. 6.5.

Для случая, когда возможно использование в ПМВ01П БИС ПЗУ емкостью 2 Кбайта, разработаны алгоритмы проверки и система ТПСА, которая целиком хранится в ПЗУ ПМВ01 и занимает приблизительно 1,5 Кбайта памяти. Алгоритмы проверки и системы ТПСА разработаны на базе метода расширяющихся областей. Для сокращения ядра оборудования выбран аппаратно-программный способ выбора определенной ТПСА внутри системы ТПСА без использования диалогового режима работы. Выбор заданной программы производится путем программного анализа признаков, устанавливаемых с помощью проводных перемычек.

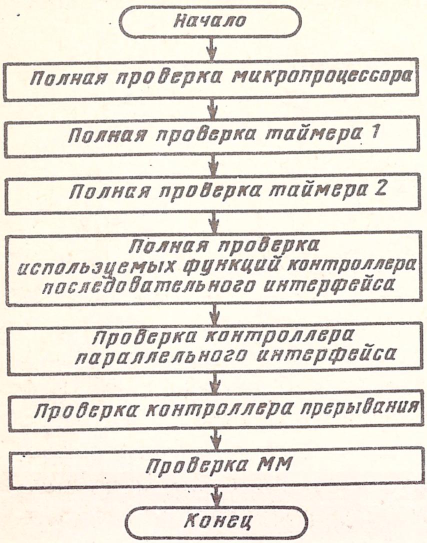


Рис. 6.5. Блок-схема алгоритма поиска неисправности в ПМВ01П под управлением ТПСА из ОЗУ

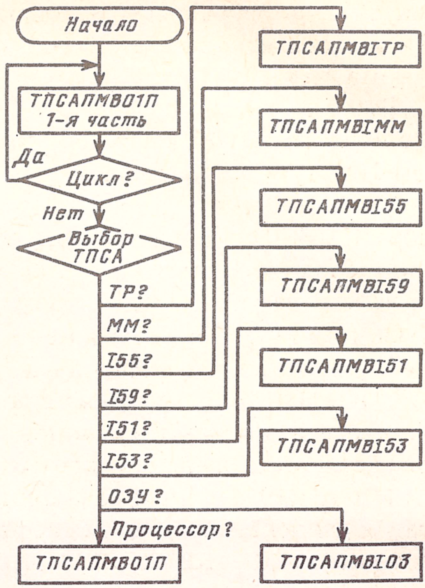


Рис. 6.6. Блок-схема системы ТПСА ПМВ01П

Система ТПСА состоит из:

ТПСАПМВ01П – ТПСА проверки микропроцессора;
ТПСАПМВ103 – ТПСА для проверки ОЗУ;
ТПСАПМВ153 – ТПСА таймеров;
ТПСАПМВ151 – ТПСА контроллера последовательного интерфейса;
ТПСАПМВ155 – ТПСА контроллера параллельного интерфейса;
ТПСАПМВ159 – ТПСА контроллера прерывания;
ТПСАПМВ1ТР – ТПСА программно-управляемых триггеров;
ТПСАПМВ1ММ – ТПСА межмашинной магистрали.

Блок-схема системы ТПСАПМВ01 приведена на рис. 6.6. Система функционально состоит из двух частей. Первая часть программы предназначена для стимуляции ядра микропроцессора и ПМВ01. В ядро устройства и микропроцессора входит следующее оборудование и команды: магистраль данных, схема и магистраль управления, внутримашинная магистраль, команды, обеспечивающие работу перечисленных схем, некоторые комнады переходов и команды группы RESTART. В этой же части ТПСАПМВ01П произво-

дится анализ условий передачи управления и переход в зависимости от условия к той или иной ТПСА или же программа зацикливается.

Вторая часть программы ТПСАПМВ01П осуществляет полную проверку системы команд, за исключением команды HALT, которая проверяется в ТПСА ПМВ159.

ТПСАПМВ103 предназначена для стимуляции ОЗУ процессора ПМВ01П. Программа выполнена с использованием модифицированного метода "Марш". ТПСАПМВ103 ориентирована на работу с ОЗУ, размещенном в диапазоне адресов 8000_{16}– $BFFF_{16}$. При изменении адресации ОЗУ программу необходимо настроить на новый диапазон адресов.

ТПСАПМВ153 предназначена для стимуляции таймера устройства, который выполнен на базе БИС КР580ВИ53. ТПСА осуществляет проверку работы таймера во всех шести режимах.

ТПСАПМВ151 предназначена для стимуляции контроллера последовательного интерфейса, выполненного с использованием БИС КР580ВВ51. ТПСА проверяет работу контроллера только в асинхронном режиме при скорости передачи 9600 бод. Осуществляется проверка работоспособности всех трех каналов, при этом входные и выходные тракты каждого канала замкнуты между собой с помощью перемычки. Проверка производится в режиме сдвига единицы на фоне нуля и нуля на фоне единиц, вывода кодов: 0000, АА55, 55АА.

ТПСАПМВ159 предназначена для стимуляции контроллера прерывания (КП). Запросы на прерывания формируются счетчиками таймеров К580ИК53. Контроллер прерывания инициализирован в режиме главного с подчиненным. Обработка прерываний КП осуществляется вначале в режиме "Опрос", а затем в режиме "Циклический приоритет", в котором одновременно производится проверка правильности выполнения и команды микропроцессора HALT (останов).

ТПСАПМВ155 предназначена для стимуляции контроллера параллельного интерфейса. Осуществляется проверка контроллера в режиме сдвига единицы на фоне нуля. Контроллер стимулируется в нулевом режиме. Порты А1, С1, А2, С2 инициализированы на ввод, а порты В1, В2 – на вывод. При этом информация из порта В1 передается в порты А2, С2, а из порта В2 – в А1, С1.

ТПСАПМВ1ММ предназначена для стимуляции схем ПМВ01, с помощью которых осуществляется обмен информацией

через межмашинную магистраль и самой магистрали. С помощью этой программы производится проверка шин данных и управления магистрали (шина адреса проверена в автономном режиме). Проверка шин данных производится с использованием вывода сдвигающейся единицы на фоне нулей.

ТПСА ПМВ1ТР предназначена для стимуляции программно-управляемых триггеров ПМВ01, адресуемых как порты ввода-вывода.

Контроль правильности хранения программы в ПЗУ перед началом поиска неисправности осуществляется путем проверки совпадения сигнатур, снятых с БИС, содержащей ТПСА ПМВ01, с ее эталонным значением. Проверка производится в автономном режиме.

Поиск неисправности с использованием системы ТПСАПМВ01 можно производить, используя метод расширяющихся областей, последовательно проверяя схемы устройства по алгоритму, представленному на рис. 6.6, а можно сократить объем проверки, сразу приступив к поиску неисправности в подозреваемом узле, который был определен при тестировании устройства ПМВ01П или неисправность которого была выявлена по неправильной работе рабочей программы. Переход к стимуляции необходимого узла осуществляется описанным выше способом. Поиск неисправности осуществляется с использованием контрольных точек, количество которых колеблется от 8 до 20 для различных узлов ПМВ01П.

Наиболее сложна локализация неисправности в той части схемы, которая стимулируется первой частью ТПСАПМВ01П. Сложность поиска неисправности заключается в том, что в проверку включается одновременно достаточно большой объем оборудования, которое влияет на правильность работы устройства в целом, а именно: шина данных, микропроцессор, непроверенная в автономном режиме часть регистра команд. Поэтому для осуществления локализации неисправности в этих схемах для ТПСАПМВ01П разработаны дополнительно две таблицы эталонных сигнатур, которые используются при невозможности локализации по основным двум таблицам (как правило, для каждой из ТПСА составлялись две таблицы эталонных сигнатур: одна для команд чтения или ввода, а другая для записи или вывода).

Стимуляция проверяемой части схемы для дополнительных таблиц аппаратно-программная. В качестве программы

используется та же часть ТПСАПМВ01П, с одновременным использованием принудительной циклической генерации сигнала "Исх. пол", который периодически устанавливает микропроцессор и все устройство в целом в исходное состояние, заставляя микропроцессор начинать каждый раз исполнение программы с адреса 0000_{16}. В качестве источника циклического сигнала выбран счетчик регенерации ОЗУ, который не сбрасывается по сигналу "Исх. пол". При этом длительность работы микропроцессора составляет 12 машинных циклов. Этот же сигнал используется для формирования ОИ в качестве сигнала "Пуск" и "Стоп" СЛ01П. Длительность же ОИ для этих двух таблиц различна.

В первой таблице определяются работоспособность шины данных и правильность генерации адреса микропроцессором при исполнении команды безусловного перехода. Поэтому с помощью режима сдвига конца ОИ выбирается такая длительность ОИ, которая должна обеспечить выполнение первых команд ТПСА, включая команду безусловного перехода. В качестве сигнала СИ СЛ01П нельзя использовать применяемый обычно сигнал "Прием" микропроцессора, поскольку в момент действия сигнала "Исх. пол" он не вырабатывается. По этой причине в качестве СИ выбран непрерывно вырабатываемый сигнал ГИ2. Однако из-за того, что во время действия ГИ2 шины данных и адреса микропроцессора согласно временной диаграмме могут находиться в неопределенном состоянии, прием данных в СЛ01П дополнительно стробируется сигналом "Прием", который используется как сигнал "Ожидание" СЛ01П. Созданный таким образом режим работы системы ПМВ01П, СЛ01П позволяет выделить и проверить правильность выполнения первых четырех команд и локализовать неисправность.

Второе ОИ для следующей таблицы предназначено для локализации отказа, связанного с отсутствием генерации сигналов "Прер" и триггера *Т* "Захв. М", используемых для здания СИ в основной таблице эталонных сигнатур, если неисправность в первой дополнительной таблице не обнаружена. Здесь также производится сдвиг конца ОИ до стимуляции программно-управляемого триггера *Т* "Захв. М".

Таким образом, введенные в СЛ01П дополнительные возможности позволяют успешно выделять интересующие наладчика сигналы и интервалы в ОИ.

Поиск неисправностей в остальных частях схемы ПМВ01П производится по заданным в таблицах алгоритмам. Однако необходимо отметить, что в процессе разработки системы диагностирования ПМВ01П были выявлены некоторые особенности работы ряда БИС серии К580, а именно: таймеров КР580ВМ53, контроллеров прерывания КР580ВН59.

ТПСА53 производит стимуляцию работы таймера во всех режимах, не производится только проверка его работы с сигналами Gate 0 – Gate 2, поскольку в базовой конфигурации ПМВ01П такая работа не предусмотрена. В процессе разработки была выявлена нестабильность работы БИС в режиме опроса с защелкиванием, причем каждая схема в отдельности работает стабильно, однако от одной БИС к другой на одной и той же плате ПМВ01П генерируются различные сигнатуры. Проведенный анализ на выборке приблизительно из 50 различных БИС, взятых из разных партий различных лет выпуска, показал, что разные БИС на одной и той же программе в одной и той же палате в этом режиме дают различные значения в младшем байте считываемого духбайтного числа. Погрешность выявлена в четырех младших разрядах. При этом определено, что все БИС разбиваются на четыре группы. Исследование БИС I8253 показало, что эти схемы ведут себя аналогично.

Описанная погрешность возникает при работе с входным синхроимпульсом высокой частоты порядка 1 МГц. На низких частотах (менее 500 кГц) явление не обнаружено. Поэтому программистам, очевидно, следует с большой осторожностью применять этот режим, поскольку от устройства к устройству или после замены одной БИС на другую считываемые значения могут изменяться. Второй особенностью этой БИС и ее аналога является работа в третьем режиме при занесении в счетчик числа 0003Н. По приведенному в описаниях порядку работы в этом режиме схема должна генерировать непрерывную последовательность из двух импульсов: первый – длиной в два и второй – в один входной синхроимпульс, однако формируемая последовательность импульсов на самом деле равна двум и 65535 синхроимпульсов. Эта особенность таймеров также должна учитываться программистами.

Контроллер прерывания БИС КР580ВН59 работает некорректно в режиме "Опрос", когда он используется как под-

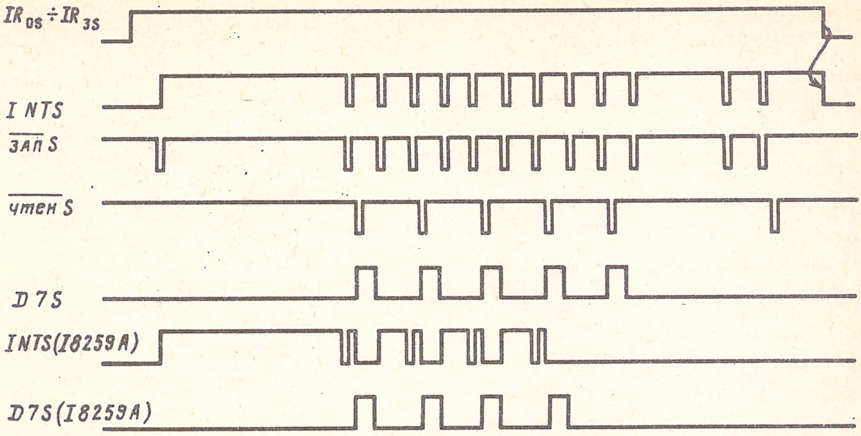


Рис. 6.7. Временная диаграмма работы БИС КР580ВН59

чиненный контроллер совместно с ведущим. Контроллер прерывания инициализировался на прием по фронту восьми запросов прерывания – четырех для ведущего контроллера и четырех для подчиненного. Запросы прерывания генерировались с помощью таймеров, которые устанавливали практически одновременно четыре выходных сигнала, поступающих на входы ЗПР обеих БИС. Режим "Опрос" инициализировался тремя командами: командой вывода, устанавливающей режим "Опрос"; командой ввода, во время которой в процессор передается слово состояния из БИС КР580ВМ59, содержащее признак наличия обрабатываемого запроса прерывания и номер входа $ЗПР_i$, на котором находится обрабатываемой в данный момент запрос, эта же команда должна рассматриваться БИС КР580ВН59 как сигнал подтверждения прерывания; командой вывода для окончания обработки прерывания. На рис. 6.7 приведена временная диаграмма работы БИС КР580ВН59 и I8259A при выполнении описанной программы обработки четырех запросов прерывания подчиненным контроллером прерывания. Как видно из временной диаграммы, БИС КР580ВН59 не осуществляет гашения регистра *IR*, поэтому в микропроцессор постоянно передается код запроса наиболее старшего прерывания. Если же запрос прерывания сразу после его обработки снимается, то работа в режиме "Опрос"

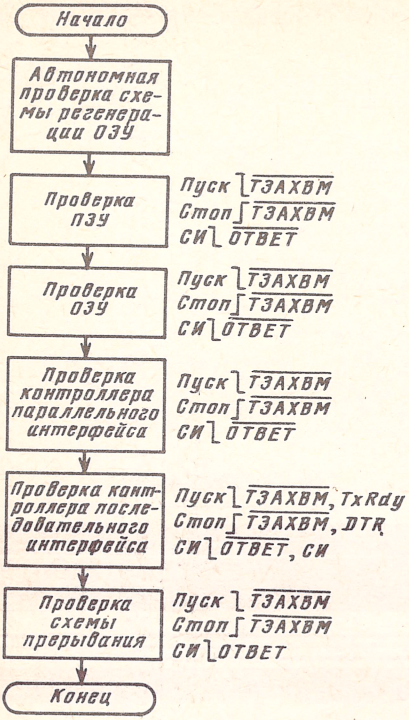


Рис. 6.8. Блок-схема проверки МВ01П

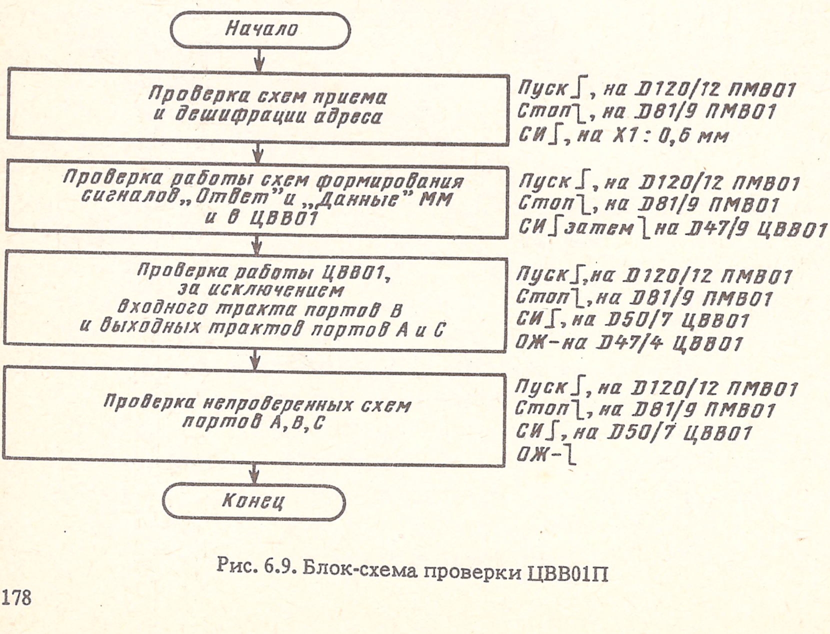


Рис. 6.9. Блок-схема проверки ЦВВ01П

может выполняться. Программист должен осуществлять работу с БИС КР580ВН59 в описанном режиме очень аккуратно.

Устройства расширения памяти МВ01П и цифрового ввода-вывода ЦВВ01П разрабатывались без учета требований сигнатурного анализа. Тем не менее для них созданы системы диагностирования с использованием этого метода. Блок-схемы их проверки представлены на рис. 6.8, 6.9.

6.3. ОРГАНИЗАЦИЯ СИСТЕМЫ ДИАГНОСТИРОВАНИЯ КОМПЛЕКСА ЦИФРОВОГО УПРАВЛЕНИЯ

Разработку системы диагностирования МПК рассмотрим на примере некоторого комплекса цифрового управления (КЦУ).

КЦУ (рис. 6.10) предназначен для приема, передачи и обработки цифровой информации с объектов, в которых используется цифровая система управления. Он включает в себя: ПМВ01П, МВ01П, ЦВВ01П, устройство связи с магнитофоном УСМП, телетайп РТА7Б, блок микросистемы БМ01П, блок питания БП01П.

Поскольку в КЦУ возможен запуск функциональных тестов, то в качестве основы выбирается алгоритм поиска неисправности с предварительным определением неисправного устройства, описанный в § 4.2. Дальнейшее определение алгоритма производится по методике, описанной там же.

Составим модель МПК КЦУ в виде графа $G(E^1, U^1)$ (рис. 6.11). Каждой вершине $e_i \in E$ и $q_j \subset Q \in E$, где $i = 1,7$;

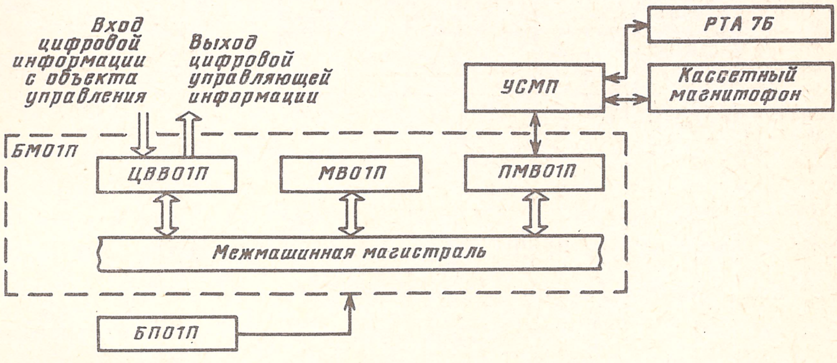


Рис. 6.10. Структурная схема МПК КЦУ

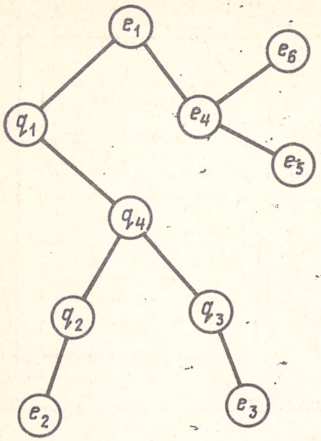


Рис. 6.11. Граф $G(E^1, U^1)$ КЦУ

$j = \overline{1,4}$ поставлены в соответствие устройства: e_1 – ПМВ01П; e_2 – МВ01П; e_3 – ЦВВ01П; e_4 – УСМ; e_5 – РТА7Б; e_6 – кассетный магнитофон; q_1 – блок связи с ММ в ПМВ01П; q_2 – блок связи с ММ в МВ01П; q_3 – блок связи в ЦВВ01П; q_4 – ММ.

Поскольку в КЦУ входит только одно ЦПУ – ПМВ01П и согласно § 4.2 оно должно входить в диагностирующую подсистему, то будем считать диагностирующую подсистему определенной и состоящей из ПМВ01П, УСМ, РТА7Б. Объединим вершины $e^1_1, e^1_4, e^1_5, e^1_6$ в одну вершину $e^1_я$ и построим на базе полученного подграфа $G^1_2(E^1_1, U)$ граф $G^3_2(E^1_1, U)$, который зададим согласно § 4.2 в виде матрицы смежностей (табл. 6.1).

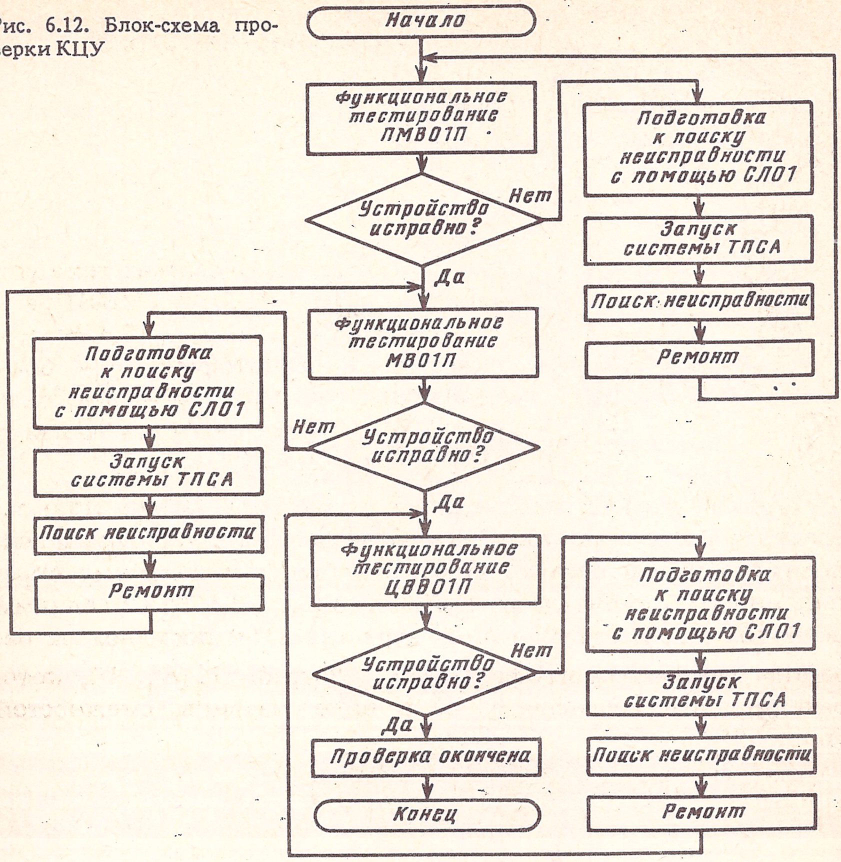
Таблица 6.1

	$e^1_я$	e^1_2	e^1_3
$e^1_я$		1	1
e^1_2			
e^1_3			

$\hat{Г}e^1_я$	$\hat{Г}e^1_2$	$\hat{Г}e^1_3$
0	X	X
1	0	X
X	1	0

Определим из табл. 6.1 транзитивные замыкания относительно вершин $e^1_я$, e^1_2, e^1_3. Для получения весового коэффициента каждой вершины произведем сложение чисел, полученных в $\hat{Г}e^1_i$ для вершин $e^1_я$, e^1_2, e^1_3 и соответственно получим 0, 1, 1. Таким образом, согласно методике, изложенной в § 4.2, первой проверяется вершина $e^1_я$, затем e^1_2

Рис. 6.12. Блок-схема проверки КЦУ



и, наконец, e_3^1. Подставим названия устройств, соответствующих каждой из вершин, в алгоритм рис. 4.9. В результате подстановки получим алгоритм поиска неисправности в КЦУ (рис. 6.12) с использованием сигнатурного анализа.

Для определения алгоритма проверки каждого устройства необходимо при его создании учитывать те изменения в устройствах, которые вносит разработчик МПК. Центральное процессорное устройство ПМВ01П реконфигурируется в части системы прерывания. Поэтому алгоритм проверки для базовой конфигурации устройства в основном сохраняется. Изменение производится только в проверке системы прерывания. Для проверки реконфигурированной системы прерывания необходимо на соответствующие контакты сиг-

налов запросов прерывания подать сигналы прерывания и при этом снять новые эталонные сигнатуры.

Устройства МВ01П и ЦВВ01П не изменяются, и поэтому они проверяются с использованием своих систем диагностирования в базовой конфигурации.

Глава 7

ЛОГИЧЕСКИЕ АНАЛИЗАТОРЫ И ЗАДАЧИ ОТЛАДКИ МПС

7.1. ОБЩИЕ СВЕДЕНИЯ О ЛОГИЧЕСКИХ АНАЛИЗАТОРАХ КАК СРЕДСТВЕ ДИАГНОСТИРОВАНИЯ И ОТЛАДКИ МПС

Процесс диагностирования и отладки с использованием ЛА. Появление логических анализаторов (ЛА) обеспечило качественно новый подход к отладке и диагностированию средств вычислительной техники. Анализаторы представляют в распоряжение инженеров метод логического анализа состояния объекта диагностирования и отладки и позволяют решать задачи, недоступные другим традиционным средствам диагностирования, работающим во временно́й области. Логические анализаторы, автоматизируя этапы регистрации и отображения информации о реальных процессах в отлаживаемом объекте, обеспечивают значительное сокращение времени отладки (в 10–20 раз). Особенно значительное сокращение времени отладки достигается для МПС, характеризующихся малым числом контрольных точек, высокой степенью интеграции составляющих компонентов, высоким быстродействием, неразделимостью аппаратуры и программного обеспечения, наличием шинной архитектуры и другими особенностями. Это достигается за счет свойств, отсутствующих в ранее существовавших средствах, таких, как осциллографы, вольтметры, генераторы сигналов, триггерные ловушки. Новые свойства следующие: регистрация последовательности логических сигналов одновременно и синхронно во многих точках устройства и на протяжении значительного временно́го интервала в однократном и периодическом режимах; регистрация последовательности состояний в связи с редкими и однократными событиями (управляющими сигналами, командами) и возможность

селективного сбора; регистрация состояния контрольных точек в интервале времени, перемежающихся выбранному, либо после выбранного, а также до и после выбранного; возможность регистрации перемежающихся неисправностей; асинхронная регистрация.

Логические анализаторы обладают возможностью оперативного отображения зарегистрированной информации в различных режимах и форматах, удобных для принятия решения оператором и не отличающихся от принятых форм представления информации.

Такие возможности обеспечиваются использованием методов запуска и синхронизации логических анализаторов, причем запуск обеспечивает желаемую точку отсчета, а синхронизация – запись нужной информации.

Режимы отображения информации характеризуются прежде всего тем, что они не зависят от режимов регистрации. Индикация может длиться как угодно долго, при этом оператор может менять режимы и формы отображения, использовать различные методы обработки, такие, как сравнение с эталонными значениями (предварительно введенными в логические анализаторы), поиск заданного слова, вычисление контрольных сумм массивов данных.

Применяются логические анализаторы на стадиях разработки, производства и эксплуатации МПС в различных организационно-технологических схемах.

Организация диагностирования предполагает выполнение следующих этапов:

изучение нормативно-технической документации и определение правильного функционирования ОД в пределах, установленных документацией;

регистрация и отображение с помощью логических анализаторов процесса преобразования информации в ОД;

анализ полученной информации и определение технического состояния.

Для работы с логическими анализаторами оператор должен обладать определенным уровнем квалификации. Для снижения квалификационных требований могут применяться различные подсказки, справочная информация.

Подготовительные операции включают анализ технической документации, выбор характеристик ЛА, построение алгоритма проверок, выбор контрольных точек, определе-

ние эталонов, задание режимов работы ЛА и ОД, подключение ЛА к ОД.

Рабочие операции состоят в анализе информационного потока системами запуска ЛА, синхронизации и регистрации диагностической информации, генерации стимулирующих воздействий.

Операции обработки: отображение диагностической информации, преобразование, измерение, сравнение, поиск, принятие решения, документирование, связь с диагностическим центром (для консультаций персонала). Каждая операция характеризуется временем проведения, качеством выполнения, степенью автоматизации.

Если рабочие операции и операции обработки являются автоматическими, то подготовительные не только не автоматизированы, но и мало формализованы.

У современных систем время подготовки достигает 30 % и зависит от контролепригодности ОД, применяемых средств подключения, уровня квалификации персонала.

Время локализации места неисправности зависит в основном от времени подготовки и времени на анализ и принятие решения.

Необходимо отметить, что во многом организация процесса диагностирования и отладки методом логического анализа совпадает с организацией по методу сигнатурного анализа: это выбор контрольных точек, составление тестовых и стимулирующих воздействий, их анализ и т.д. Эти вопросы подробно изложены в гл. 4 и 5. В настоящей главе будут рассматриваться вопросы, свойственные преимущественно методу логического анализа – особенности запуска и синхронизации, последовательные алгоритмы отладки, применение моделирования для получения эталонных значений в контрольных точках, диагностирование перемежающихся неисправностей.

Особенности применения ЛА для МПС. Как известно, любая МПС состоит из совокупности определенных блоков, таких, как процессор, оперативная память, постоянная память, интерфейсы устройств ввода-вывода, магистральные шины управления, адреса данных.

Организация МПС с помощью магистральных шин, их многофункциональность при передаче различных элементов, наличие аналоговых устройств в их составе обусловли-

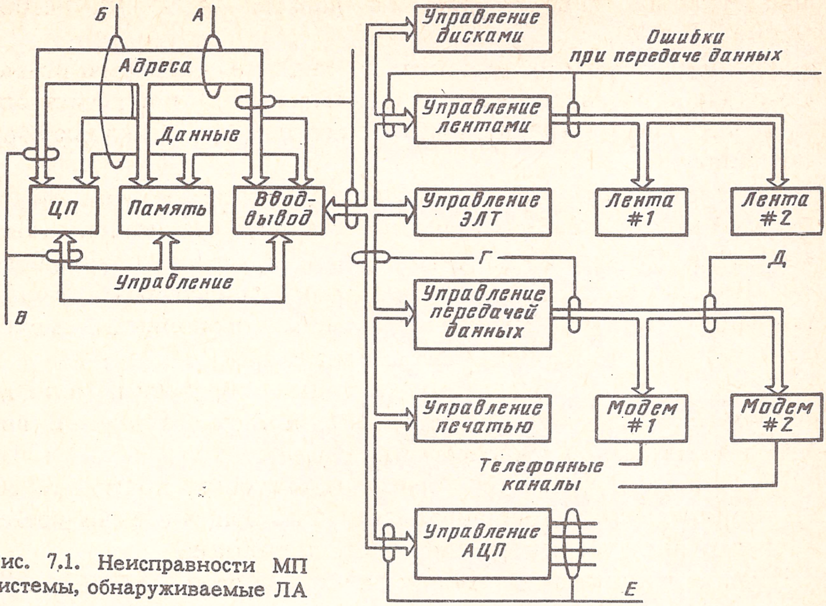


Рис. 7.1. Неисправности МП системы, обнаруживаемые ЛА

вают особенности отладки. В частности, важную роль играет шина, с которой можно снять информацию о состоянии того или иного блока.

Таким образом входы ЛА необходимо подключать в первую очередь к шинам. Затем определяется часть МПС, где происходит сбой путем регистрации и анализа данных и адреса на шине.

Рассмотрим основные неисправности в МПС. К ним относятся кратковременные импульсные помехи (составляющие, по оценкам, 90 % всех отказов), шумы, в том числе связанные с программой ошибки, обусловленные последовательностью импульсов синхронизации, влиянием нестабильности синхронизации на программу (рис. 7.1).

Блоки *А–Е*, показанные на рис. 7.1, содержат:

блок *А*: 1) ошибки алгоритмов; 2) сбой в цикле; 3) битовые искажения;

блок *Б*: 1) ошибки при передаче данных; 2) ошибки ввода-вывода; 3) ошибки в алгоритмах; 4) битовые искажения;

блок *В*: 1) помехи, вызванные программой; 2) влияние сигналов управления; 3) влияние изменения сигналов синхрони-

зации на программу; 4) неверная передача управления; 5) битовые искажения; 6) "зависание" системы; 7) превышение времени работы;

блок *Г*: ошибки передачи;

блок *Д*: 1) протоколы; 2) ошибки управления; 3) ошибки при передаче данных;

блок *Е*: ошибки при аналого-цифровых и цифро-аналоговых преобразованиях.

Ошибки программного типа: зацикливание, превышение времени выполнения команд, потеря программ, ошибки алгоритма, ошибки в передаче данных, ошибки установления связи между устройствами и др. Для определения этих ошибок необходимо специальным образом подбирать места подключения к контрольным точкам синхронизации и способы запуска. Входы данных, синхронизации и признаков ЛА подключаются по-разному.

Входы данных позволяют снять интересующую оператора диагностическую информацию.

Входы синхронизации можно подключать к самым разнообразным сигналам: к тактовому сигналу МПС; к любому асинхронному (синхронному) сигналу, например по сигналам из устройства управления; к нескольким фазам многофазной синхронизации МПС.

Входы признаков подключают к таким сигналам, при наличии которых должна быть зарегистрирована в памяти ЛА информация, присутствующая на входах данных. Эти сигналы являются дополнительным условием запуска. Таким сигналом является, например, считывание = запись в память, сигналы подтверждения установления связи, сигнал помехи.

Таким образом, контрольными точками на системном уровне являются сигнальные линии шин адреса, данных и управления, в том числе внутренних и периферийных.

Схемному уровню соответствуют контрольные точки на ТЭЗах и отдельных микросхемах. При проверке отдельных функциональных блоков входы ЛА рекомендуется подключать к кодовым выходам этих блоков, а вход синхронизации – ко входу синхронизации этих блоков.

Таким образом, для МПС на системном уровне можно использовать следующие режимы измерений: правильность выполнения команд, правильность выдачи сигналов управления, правильность передач по шинам.

Современные ЛА имеют несколько режимов, специально приспособленных для диагностирования перемежающихся неисправностей – сравнения таблиц, обнаружения коротких импульсов ("глич"-импульсов), сбора данных с разделением памяти.

Режим сравнения таблиц заключается в следующем. Первоначально в ЛА устанавливаются условия запуска для ввода эталонных данных в основное ОЗУ (A), например, от заведомо исправного устройства или типового элемента замены. Затем эти данные переписываются в опорное ОЗУ (B). Это осуществляет пользователь путем нажатия клавиши на передней панели. Затем ЛА переводится в режим $A \neq B$, остановки по несовпадению содержимого A и B, в котором используются те же условия запуска, что и раньше. Как только исходные условия запуска появляются на входе ЛА, то ЛА записывает данные в основное ОЗУ (A) и сравнивает с данными из опорного ОЗУ (B). Если содержимое A и B совпадает, то выполняются дальнейшие диагностические процедуры. Если в каком-либо из очередных циклов содержимое A и B не совпадает, то это событие классифицируется как перемежающийся отказ, вырабатывается сигнал $A \neq B$ и ЛА повторно запускается, при этом прекращается сбор очередных данных. Затем оператор с помощью дисплейного режима $A \oplus B$ задает режим табличного сравнения эталонных данных с новыми данными из ОЗУ (A), в результате чего определяются место и момент возникновения перемежающейся неисправности.

Дальнейшим усовершенствованием этого режима является метод, суть которого состоит в сравнении информационных массивов основной (рабочей) памяти A и опорного (эталонного) ОЗУ (B), которое происходит постоянно по каждому очередному записываемому слову в темпе поступления входных данных. При этом режиме обеспечивается более высокая вероятность обнаружения динамических неисправностей. Для реализации этих режимов ЛА содержит специальные компараторы, регистры и схемы управления.

Обнаружение коротких помех ("глич"-импульсов) производится путем фиксации их ЛА специальными схемами (триггерами-защелками) и последующего запуска сбора данных. Триггеры-защелки растягивают импульс до одного

интервала квантования, затем он воспроизводится в следующем такте в виде импульса минимальной ширины.

Во втором методе записи детектор помех работает параллельно запускам и тем самым отделяет импульс помех от выбранных данных. Этот метод удобнее, так как позволяет сигнал помехи использовать непосредственно для запуска ЛА или в функции ИЛИ со словом запуска.

В режиме сбора с разделением памяти для каждого события запуска в память ЛА записывается некоторое, заранее установленное оператором число слов данных до заполнения памяти. Таким образом, за один цикл сбора данных можно многократно проверять место в потоке данных, где имеются перемежающиеся неисправности.

Архитектура ЛА. Микропроцессорные анализаторы. В соответствии с функциональным назначением ЛА состоит из запоминающего устройства, устройства преобразования диагностической информации, ввода, вывода, анализатора информации, пульта управления.

Запоминающее устройство (ЗУ) анализатора служит для приема и хранения информации, получаемой в процессе диагностирования. Емкость ЗУ определяется количеством входных каналов.

Устройство преобразования диагностической информации представляет собой ЭВМ, микропроцессорный контроллер или автомат с жесткой логикой для преобразования информации, хранимой в ЗУ.

Устройство ввода осуществляет преобразование информации, поступающей на вход ЛА в двоичный код, методом сравнения уровней напряжения сигналов, которые поступают на вход с уровнями, установленными в ЛА.

Устройство вывода представляет собой дисплей для отображения диагностической информации после ее обработки устройством преобразования служебной информации. В современных приборах имеются устройства вывода для документирования и передачи информации в линии связи.

Пульт управления служит для задания режимов работы ЛА оператором.

Устройство анализатора информации выдает сигналы управления работой ЛА при совпадении кодов на выходе устройства ввода с кодами, установленными оператором. Основные модули ЛА показаны на рис. 7.2.

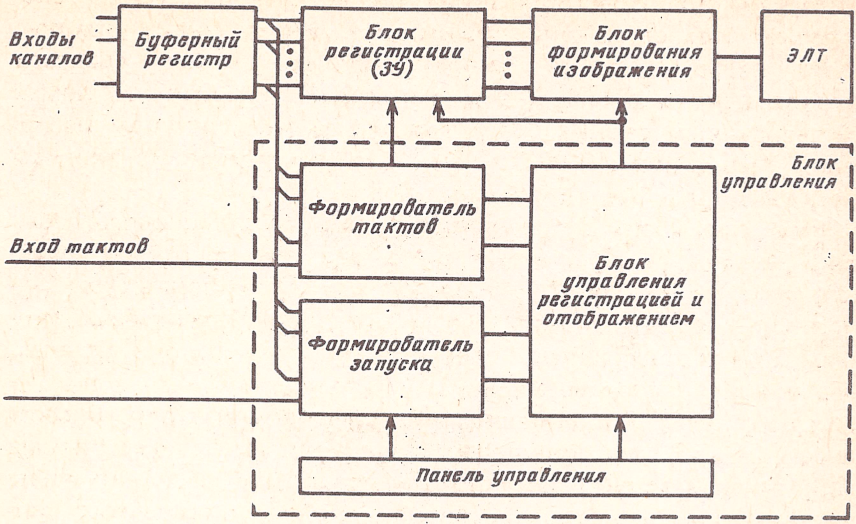


Рис. 7.2. Структурная схема логического анализатора

Анализатор "разворачивает" состояние объекта диагностирования в дискретном времени, задаваемом внутренним или внешним генератором. Генератор, задавая последовательность импульсов, определяет моменты времени, в которые регистрируется состояние ОД в виде вектора, число элементов которого соответствует числу выходов ЛА. Каждый элемент заносится в память ЛА в виде двоичной переменной. Значение переменной определяется входным компаратором уровня ЛА по результату сравнения потенциала напряжения в контрольной точке ОД с уровнем, заданным оператором.

Записанная в виде наборов векторов информация отображается на дисплее.

Основными характеристиками ЛА являются: число каналов, глубина памяти (бит/канал), максимальная частота регистрации, способ запуска регистрации и индикации.

По функциональному назначению ЛА разделяют на анализаторы логических временных последовательностей (АЛВП), использующих асинхронную регистрацию, и анализаторы логических состояний (АЛС) с синхронной регистрацией. Частота регистрации АЛВП должна быть хотя бы на поря-

док выше тактовой частоты современных МПС. АЛВП представляют собой более сложные приборы, чем АЛС.

По применению различают ЛА общего и специального назначения. Логические анализаторы специального назначения различают для контроля шин цифровых систем, для микропроцессорных систем, для программного обеспечения. С точки зрения пользователя важны такие признаки, как условия применения (сервисные или лабораторные), степень автономности (встроенные, внешние), конструкция (одноплатные и многоплатные), характеристики управления, методы отображения информации и др.

Можно выделить три направления развития и использования ЛА:

специализированные приборы, ориентированные на соответствующий этап существования средств вычислительной техники: для производства, эксплуатации и наладки аппаратуры и программного обеспечения, а также ориентированные на определенный тип дискретных устройств (интерфейсы ЗУ, микропроцессоры, шины и т.п.);

мощные универсальные приборы для разработки и отладки опытных образцов средств вычислительной техники;

модульные системы, управляемые универсальными мини- и микроЭВМ, в том числе ПЭВМ.

Последнее направление является одним из наиболее перспективных с учетом широкого распространения ПЭВМ, так как обеспечивает высокие показатели процесса диагностирования в сочетании с универсальностью и хорошими сервисными возможностями.

Микропроцессорные анализаторы. Широкое применение микропроцессоров привело к созданию целого ряда специализированных логических анализаторов, ориентированных на разработку микропроцессорных систем.

Одним из таких анализаторов является прибор VP-3662A фирмы Matsustita, имеющей 32 канала при частоте выборки 20 МГц, или 16 каналов при частоте выборки 100 МГц.

Прибор анализирует процессы в линиях передачи данных, а также предназначен для отладки 8- и 16-разрядных микропроцессорных систем.

Анализатор имеет набор плат для работы с системами на основе микропроцессоров I8080, I8085, Z80, 6802, 6809, I8088, I8086, Z8001, Z8002 и 68000.

Прибор функционирует в реальном масштабе времени и отображает мнемонические обозначения, дизассемблируя машинные коды микропроцессора. Предусмотрен режим обнаружения помеховых импульсов по 32 каналам. На случай потери напряжения память снабжена резервным батарейным питанием.

Некоторые фирмы–изготовители диагностической аппаратуры для усиления конкурентоспособности своей продукции начали выпуск контрольно-измерительных приборов на основе персональных компьютеров.

Так фирма Omni Logic приступила к выпуску простого в обращении, управляемого с помощью меню команд анализатора логических состояний и временных отношений Omni II. Прибор реализован на базе популярного персонального компьютера, работающего под управлением операционной системы CP/M. Анализатор может регистрировать до 1000 отсчетов цифрового сигнала по 16 каналам или 330 отсчетов по 8 каналам в зависимости от поставляемого варианта. Предусмотрена возможность фиксации коротких помеховых импульсов с минимальной длительностью 10 нс по 4 каналам. Система работает с внутренним или внешним тактовым сигналом частотой не более 20 МГц, имея три режима запуска по условиям И, ИЛИ и НЕ относительно данных или помеховых импульсов, причем точка запуска может размещаться на любой глубине в памяти отсчетов входного сигнала.

После регистрации данных прибор Omni II позволяет отобразить стандартные временные диаграммы по 16 каналам или диаграммы фронтов, облегчающие идентификацию моментов смены состояний.

Дальнейшее совершенствование логических анализаторов идет по пути интеграции свойств различных контрольно-измерительных приборов, совершенствования системной архитектуры и обеспечения универсальным комплектом программного обеспечения.

Примером такого прибора может служить многофункциональный тестер № 2610 фирмы Genrad, сочетающий в себе свойства цифрового мультиметра, логического анализатора, анализатора сигналов, системы проектирования микрокомпьютеров и осциллографа.

Универсальные ЛА начали активно вытеснять на рынке диагностической аппаратуры анализаторы, специализирующиеся только на отладке программных или аппаратных средств.

Наиболее совершенные универсальные анализаторы представляют собой приборы, позволяющие работать на них операторам, имеющим различный уровень квалификации и опыта. Они охватывают широкий диапазон приложений и легко адаптируются к любым рабочим условиям.

Использование приборов операторами с различным уровнем квалификации достигается созданием специального интерфейса "человек машина", позволяющего в максимально простом и ясном виде представить все функциональные средства прибора и обеспечить легкий доступ к этим средствам. Первая особенность интерфейса позволяет в минимально короткие сроки освоить предоставляемые функциональные средства, вторая предоставляет квалифицированному оператору возможность задавать режим работы прибора при помощи клавиш, избегая длительного этапа программирования.

В наиболее совершенных приборах данного класса это решается использованием операционной системы с управлением от меню команд в сочетании с дисплеем сенсорного типа. Дисплей сенсорного типа представляет собой принципиально новый тип устройства, применяемого для задания команд, выполняемых прибором. Экран данного дисплея разбивается на определенное число прямоугольных графических полей с соответствующими метками внутри. Каждое из таких прямоугольных полей представляет собой определенную команду. Вызов любого набора команд достигается простым прикосновением к соответствующим полям. По периметру экрана расположены пары светоизлучающий датчик – фотоэлектрический диод. Точка пересечения лучей датчиков, расположенных по вертикали и горизонтали, является центром графического поля, задающего одну из команд. Схема опроса диодов поочередно определяет центры полей, которыми произведено задание соответствующих команд, заносит эти данные в сдвиговый регистр, содержимое которого переписывается в хранящий регистр, где производится окончательное формирование меню команд, набранных оператором.

Рассмотрим некоторые особенности работы универсального анализатора модели 1240 фирмы Tektronix [102]. Меню команд построено по иерархическому принципу. С пульта оператора можно кнопкой задать одну из пяти основных команд, при этом на экране появляется подменю операций, соответствующих заданной основной команде. Выбор набора из предложенного числа операций осуществляется вышеописанным способом.

В процессе разработки архитектуры анализатора были расширены возможности управления воспроизведением собранных данных. Главным из них стали средства, позволяющие плавно перемещать изображения на экране – таблицы состояний по вертикали, а временные диаграммы по горизонтали. Перемещение реализовано с помощью поворотной ручки, непосредственно управляющей перемещением. При этом поворот ручки представляется в виде содержимого специального регистра, которое считывается центральным процессором и используется в качестве исходного параметра для алгоритмических программ, обеспечивающих обновление памяти изображения.

Ввиду того что данный способ перемещения изображения требует значительных накладных затрат процессора и учитывая сложность реализации функций, связанных с механизмом меню, была выбрана двухпроцессорная архитектура прибора. Задача сбора данных реализуется центральным процессором, каким является I8088, а функции воспроизведения и перемещения изображения выполняются процессором ввода-вывода, реализованным на Z80.

Управляющий процессор считывает данные из высокоскоростного ЗУ сбора данных через однобайтовые буферные регистры и запоминает их в памяти. Процессор ввода-вывода формирует изображение для воспроизведения в своей оперативной памяти, из которой оно считывается аппаратным способом для вывода на экран и перемещается по экрану при необходимости. В ходе работы процессор ввода-вывода преобразует положение управляющей ручки в четырехбайтное слово, посылаемое им управляющему процессору, после чего получает от управляющего процессора данные и формирует их в своей оперативной памяти в обновленное изображение.

Одним из главных достоинств прибора является раздельный сбор данных, т.е. способность принимать данные с ис-

пользованием двух различных частот развертки, причем это делается и совмещенно, и с корреляцией. Такая возможность удобна для систем с двухпроцессорной архитектурой.

В анализаторах с одной временной разверткой для анализа обмена данными между двумя процессорами в реальном масштабе времени используют одну из следующих двух трудоемких операций. Необходимо либо зафиксировать данные обоих процессоров асинхронно и затем просмотреть громадный объем данных для нахождения соответствующей информации, либо попытаться синтезировать общий синхросигнал, что может оказаться сложной или нерешаемой задачей.

В приборе 1240 происходит фиксация данных параллельно, с использованием двух временных разверток. Предусмотрено формирование условий запуска на основе информации, относящейся к обеим временным разверткам, так что воспроизведение результатов можно осуществить с привязкой к моменту возникновения соответствующего события.

Желая создать максимум удобств для пользователя, разработчики большое внимание уделили методам запуска работы. С этой целью условия запуска разделили на блок, опознающий одно глобальное событие, и блок, опознающий последовательные события.

Блок, опознающий глобальное событие, устанавливается в истинное состояние, когда при выборке данных происходит данное событие, после чего может инициироваться запуск необходимого режима.

Блок, опознающий последовательные события, рассчитан на последовательность из 14 событий или уровней. После определения последовательности заданных событий может осуществляться трассировка хода программы, сброс, задержка времени, переход на другой уровень.

Развитое программное обеспечение анализатора позволяет пользователю задавать с пульта число каналов, емкость буфера на канал и частоту выборки.

Сложность заключительного этапа отладки микропроцессорных систем явилась причиной создания специализированных анализаторов для этапа интеграции программных и аппаратных средств. Примером такой системы является модель 3501 фирмы Yokogawa Electric Works.

Фирма Gould производит анализатор 9616S для системы проектирования микрокомпьютеров, используемый на заключительном этапе отладки [103]. Здесь впервые совмещены функции эмулятора и логического анализатора, что позволило более точно определить влияние программного обеспечения на состояние аппаратных средств, с одной стороны, и произвести анализ воздействия специфических особенностей аппаратуры на ход выполнения программы.

7.2. СПОСОБЫ ЗАПУСКА. СИСТЕМЫ СИНХРОНИЗАЦИИ

В современных МПС информационные потоки сложны по своей структуре и соотносятся с определенными двоичными событиями, как-то: команды, данные, совокупность сигналов на управляющих линиях и др. Запуск обеспечивает желаемую точку отсчета, выделяя в потоке данных интересующее "Окно измерений" для записи информации в ЗУ и делает его видимым для пользователя. Глубина и ширина зарегистрированной информации определяются количеством входных каналов и емкостью памяти ЛА.

Наличие запуска является тем свойством, которое и делает ЛА незаменимым прибором для наладки и диагностирования.

Операция анализа информационного потока выполняется в ЛА с помощью системы, задающей момент регистрации или генерации, т.е. с помощью систем запуска. Операция анализа выполняется сравнением отдельных характеристик сигналов на входе с величинами, заданными оператором.

Признаки контролируемых сигналов, связывающих момент совпадения информации ЛА на входе с условиями, установленными оператором, могут быть различными. В связи с этим условия запуска делят на параметрические, логические и смешанные, они же определяют системы запуска.

Параметрические системы запускают ЛА при соответствии параметров сигналов, приходящих на его вход и заданных оператором. В качестве параметров могут выбираться: уровень, длительность, время нарастания сигнала, наличие сигнала (запуск по помехе). Параметрический запуск чаще всего используется для контроля сигналов ОД анализаторами временных последовательностей.

Логический запуск определяет соответствие логического значения сигналов на входе ЛА значению, заданному оператором.

Простейшим логическим запуском является комбинационный запуск по кодовому слову.

Логический запуск можно разделить на многоразовый автоматический и одноразовый автоматический.

Непосредственным является запуск, который разрешает регистрацию сразу после нажатия кнопки "Пуск" и является частным случаем комбинационного запуска.

Простой комбинационный запуск позволяет выполнить наладку аппаратуры и может быть использован при грубой отладке программ. Память ЛА при этом содержит все, что случилось до, после или близко к моменту запуска. Последовательный запуск позволяет контролировать выполнение программ, регистрировать ход программы, поскольку запускающее слово можно выбирать в точках ветвления программы.

Последовательный (многоразовый) запуск может быть с неизменными или с программируемыми начальными условиями.

Дополнением к системам запуска является смещение или задержка запуска относительно выбранных условий. Задержки могут определяться количеством импульсов синхронизации или количеством появлений кодовых слов, задаваемых последовательным или комбинационным запуском.

Комбинационный, последовательный запуски представляют собой частные случаи многоразового запуска с программируемыми условиями.

Использование сложных систем многоразового запуска позволяет: связывать логические ошибки, последствия которых проявляются через большой отрезок программы, проверять время прохождения отдельных команд, работать с многопроцессорными системами, отслеживать выполнение программы от начала до любой точки, регистрировать выполнение отдельных подпрограмм.

Смешанный запуск объединяет частично логический и параметрический.

Сложность программного обеспечения, используемого в современных МПС и имеющего множество ветвей и циклов, а также многоуровневые вложенные программы, обуслови-

ла широкое применение такого эффективного способа запуска, как запуск по последовательности кодовых слов или многоуровневый запуск. Этот вид запуска позволяет выбрать в программе, имеющей множество ветвей, один определенный путь, поскольку запуск произойдет только при соответствии порядка регистрации слов запуска заданному. Многоуровневый запуск позволяет проследить прохождение программы, например, через шесть уровней прерываний удаления от основной программы.

При многоразовом запуске в течение цикла работы ЛА проводит регистрацию несколько раз. Запуск с неизменными условиями бывает с частичным заполнением памяти ЛА и с замещением информации. Когда выполняется запуск с частичным заполнением памяти, после каждого выполнения условия запуска ЛА производит регистрацию данных около слова запуска и затем вновь переходит к поиску условий в потоке информации. После их нахождения опять выполняет регистрацию и т.д., до полного заполнения памяти. В памяти ЛА хранится информация о каждой области, в которой находятся кодовые слова. Выполняя запуск с замещением информации, ЛА после каждого нахождения условия запуска проводит регистрацию до полного заполнения памяти, затем переходит в режим сравнения и сравнивает полученную информацию с эталонной. По результатам сравнения ЛА может повторно перейти в режим регистрации и приступить к поиску условий или окончить работу.

Многоразовый запуск с программируемыми условиями является наиболее мощным средством запуска ЛА. Многоразовый запуск с программируемыми условиями позволяет проверить функционирование объекта на всем участке выполнения программ. За первым этапом поиска определенных данных (кодовых слов) в информационном потоке следует квалификация этих данных, по результатам которой ЛА осуществляет свою дальнейшую работу, т.е. когда условия запуска выполнены, в анализаторе проводится квалификация полученного условия и прибор начинает выполнять одно из нескольких следующих действий: поиск других условий запуска, регистрацию только появившегося участка, регистрацию определенного участка информационного потока и т.п.

При запуске разрядность кодового слова может быть увеличена с помощью квалификатора запуска. Под квалификатором подразумевается дополнительный сигнал, который не регистрируется в памяти анализатора, но адекватность его заданному пользователем необходима для начала регистрации. С возрастанием количества входных каналов в ЛА квалификаторы перестали использоваться для увеличения разрядности запускающей комбинации, а стали служить при работе с мультиплексными шинами для разделения адресов и данных и в мультипроцессорных системах для разделения данных от каждого процессора.

При отладке программных средств важен способ отображения зарегистрированной информации. Представление в виде временных диаграмм, используемое в АВП, удобно при отладке аппаратных средств, но непригодно при отладке программных средств. Представление логических состояний в виде 1, 0, т.е. в двоичном коде, уже можно использовать с целью отладки программных средств МПС. Недостатком двоичного формата является трудоемкость анализа информации. В настоящее время широко используется представление в кодах команд отлаживаемых микропроцессоров.

Одним из факторов, определяющих эффективность использования логических анализаторов (ЛА), является их способность регистрировать только нужную информацию, т.е. информацию, которая обеспечивает точное и быстрое прохождение по пути от проявления неисправности до установления причин ее возникновения. Такую способность дают ЛА система синхронизации и система запуска регистрации. Система запуска регистрации определяет моменты сбора информации ЛА. Система синхронизации (СС) выделяет информацию, которая регистрируется в памяти ЛА. Выбор СС является важной задачей при проведении отладки и диагностики с помощью ЛА. Конкретные СС по-разному организуют импульсы для отбора и регистрации данных. Различная организация этих импульсов позволяет делить СС на несколько видов (рис. 7.3):

1) по расположению источника синхроимпульсов СС делятся на внутреннюю, внешнюю и смешанную;

2) по взаимодействию синхросигналов, поступающих на различные синхровходы ЛА, – простую и комбинационную;

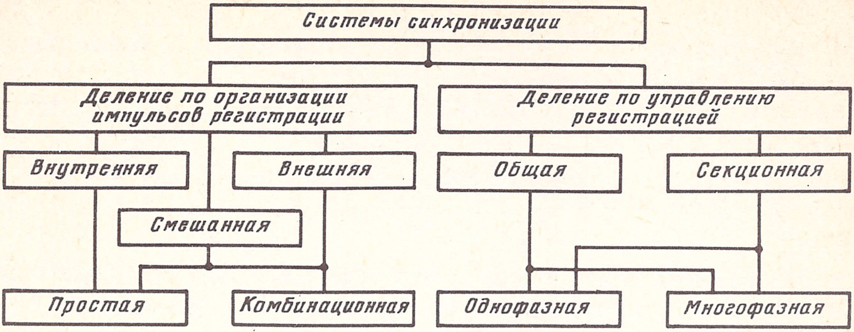


Рис. 7.3. Системы синхронизации

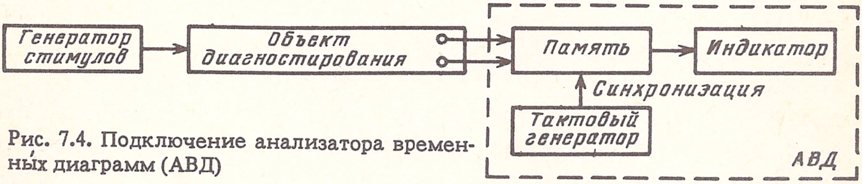


Рис. 7.4. Подключение анализатора временных диаграмм (АВД)

3) по порядку занесения информации в память ЛА – однофазную и многофазную;

4) по количеству каналов ЛА, охватываемых синхронизацией, – общую и секционную.

Каждый вид СС позволяет выполнить свой выбор информации, необходимой для решения специфичной задачи контроля и диагностики.

Внутренняя СС применяется в анализаторах временных последовательностей для регистрации данных об ОД в асинхронном режиме. При внутренней СС источник синхроимпульсов расположен внутри анализатора и не зависит от работы ОД (рис. 7.4).

Анализаторы временных последовательностей применяются для точных измерений. Погрешность выполнения измерений определяется соотношением частоты синхроимпульсов и частоты изменения информации на входе анализатора:

$$\delta = (f_и / f_с)\ 100\% \qquad (7.1)$$

где δ – погрешность измерения, %; $f_с$ – частота синхроимпульсов; $f_и$ – частота изменения информации.

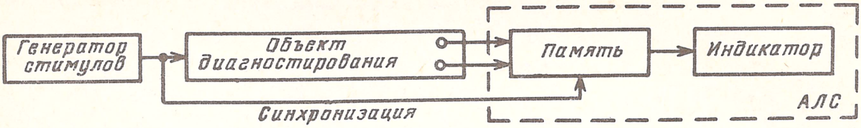


Рис. 7.5. Подключение анализатора логических состояний

Из соотношения видно, что при возрастании частоты синхронизации увеличивается точность измерений. Но поскольку по каждому импульсу синхронизации в память ЛА заносятся данные, то емкость памяти ограничивает измеряемый интервал времени, определяемый как

$$\Delta t = M/f_c,$$

где Δt – измеряемый интервал времени; M – глубина памяти ЛА, определяемая числом бит на канал; f_c – частота синхроимпульсов.

Частота внутренней синхронизации выбирается из (7.1) с учетом приемлемой погрешности измерений и измеряемого временно́го интервала. Когда частота синхроимпульсов в 10 раз больше частоты изменения информации на входе, погрешность измерений составляет 10 %, когда в 100 раз, погрешность измерений составляет 1 %. Соответственно измеренный временной интервал во втором случае меньше в 10 раз. В современных ЛА частоты синхронизации достигают значений 300 МГц (ЛА 64300 фирмы Dolch), 400 МГц (ЛА 7611 фирмы Solotron), 500 МГц (ЛА К-500 фирмы Biomution), 660 МГц (ЛА 9100 фирмы Tektronix). Асинхронная регистрация с высокой частото́й позволяет выполнять точные измерения длительности импульсов, задержки, смещения между фронтами.

При выборе внешней синхронизации для записи в память ЛА используются импульсы от ОД или генератора стимулов (рис. 7.5). Внешняя синхронизация является основной в анализаторах состояний. Можно выделить следующие виды внешней синхронизации: простую и комбинационную. Момент регистрации для обоих видов определяется по выбору оператора фронтом или срезом импульсов синхронизации. У простой синхронизации для записи используется

только один тип импульсов, у комбинационной – несколько типов импульсов.

Комбинационная синхронизация имеется у всех современных ЛА и применяется при проверках машинных циклов многотактных МП, секционных процессоров, логики работы систем управления, коммутационных шин, работы внешних устройств ЭВМ. С помощью комбинационных СС можно выделить исполнение отдельных команд, работу программ и подпрограмм. Импульсы, поступающие на разные входы комбинационной синхронизации, объединяются с помощью действий И и ИЛИ булевой алгебры и теоремы Де Моргана. Полученные таким образом импульсы называются ведущими и будут определять моменты записи в ЛА. Ведущие импульсы регистрации находятся последовательным составлением уравнений с участием двух переменных и указателя, отмечающего активный (используемый для записи) фронт или срез. Например, для двух входов C_1 и C_2, объединяемых выражением $P = \uparrow C_1 \downarrow C_2$, запись в ЛА производится по фронту C_1, когда на входе C_2 логический ноль, или по срезу C_2, когда на входе C_1 логическая единица. Для выражения $P = \uparrow C_1 + \uparrow C_2$ запись в ЛА производится по фронту синхроимпульсов C_1 или C_2, когда на другом входе имеется логический ноль. Для двух входов можно записать следующие восемь комбинаций:

$$\left.\begin{aligned}
P &= \uparrow C_1 + \uparrow C_2\\
P &= \uparrow C_1 + \downarrow C_2\\
P &= \downarrow C_1 + \uparrow C_2\\
P &= \downarrow C_1 + \downarrow C_2\\
P &= \downarrow C_1 \downarrow C_2\\
P &= \uparrow C_1 \uparrow C_2\\
P &= \downarrow C_1 \uparrow C_2\\
P &= \uparrow C_1 \downarrow C_2
\end{aligned}\right\}, \qquad (7.2)$$

Используя эти выражения как базовые, строят более сложные системы комбинационной синхронизации, например, для

семи входов:

$$P = (\uparrow C_1 + \uparrow C_2)\uparrow C_3 + (\uparrow C_4 \uparrow C_5) + (\downarrow C_6 + \downarrow C_7).$$

Частота комбинационной синхронизации определяется быстродействием входных каналов ЛА и составляет у современных анализаторов 25–50 МГц (25 МГц – ЛА 64300 фирмы Dolch; 50 МГц – ЛА К–102 фирмы Biomution).

Если необходимо выполнить диагностику многопроцессорных систем или проследить работу и взаимодействие параллельно работающих устройств, используются ЛА с секционной синхронизацией. При секционной синхронизации ЛА делится на секции, каждая из которых может выполнять регистрацию независимо от простой или комбинационной синхронизации, поступающей на свою секцию. У всех секций может быть еще и общая (ведущая) синхронизация, синхронизирующая все входы независимо от синхронизаций секций или в кобминациях с ними.

В рассмотренных видах синхронизации информация заносится непосредственно в память ЛА, поэтому они являются однофазными. В случае, когда на входе ЛА имеются промежуточные регистры, а запись в память производится через эти регистры в несколько этапов, имеет место многофазная синхронизация.

Например, при двухфазной синхронизации на первой фазе по импульсам входа C_1 информация заносится в промежуточный регистр; когда приходит синхросигнал по входу C_2, информация заносится в память со входа и из промежуточного регистра. Такая синхронизация необходима при разделении адреса и данных в мультиплексных шинах, при анализе информационных посылок в параллельных и последовательных каналах.

Возможна (например, в ЛА МА Analyst 2200) многофазная синхронизация и по одному импульсу. В этом случае по фронту или срезу импульса информация заносится в промежуточный регистр, а через интервал времени, установленный оператором, ЛА вырабатывает искусственный импульс записи и информация со входа и из промежуточного регистра заносится в память ЛА. Такая запись полезна, когда проверяются последовательные каналы передачи, где за

первым импульсом за датчиком синхронизации через определенные промежутки следует информация.

Описанные методы отбора данных для регистрации в ЛА позволяют повысить эффективность использования памяти прибора за счет предварительного отсева излишней информации. Отсев второстепенных данных, затрудняющих анализ работы ОД, позволит сократить время поиска неисправностей.

Однако не все ЛА, особенно ЛА первых поколений, обладают такими возможностями. Так, в приборе ЛА 806 807 отсутствует комбинационная синхронизация, в приборе ЛА 821 и ТСА 107 – многофазная и секционная синхронизация. В то же время они сейчас широко применяются для средств ВТ. Поэтому рационально использовать простые приставки, подключаемые на вход этих ЛА, чтобы расширить их функциональные возможности за счет введения новых видов синхронизации.

Практически может быть выполнена приставка, обеспечивающая любой рассмотренный вид синхронизации, за исключением секционной, поскольку секционная СС связана со структурой построения ЛА. Внутренняя синхронизация может быть организована в виде генератора, запускающегося от ЛА сигналом "Пуск" и подающего на вход ЛА последовательность импульсов.

Комбинационная синхронизация может быть выполнена в виде логической схемы, описываемой выражениями (7.2).

Многофазная синхронизация может быть получена путем подключения к части информационных входов ЛА регистра, в который записываются данные на первой фазе регистрации.

Если к входу синхронизации ЛА подключить через регулируемую задержку "вход – занесение в регистр", то получим многофазную синхронизацию от одного импульса. Схематично подключение приставок к ЛА представлено на рис. 7.6.

Таким образом, можно заключить, что СС, применяемые в ЛА, связаны с синхронизацией и управлением ОД, сложность которых и будет определять выбор соответствующей СС. При правильном выборе СС получают наиболее полную информацию об ОД, что и будет определять качество логического анализа.

Примеры использования ЛА для цифровых устройств. Рассмотренные выше свойства обеспечивают такие режимы

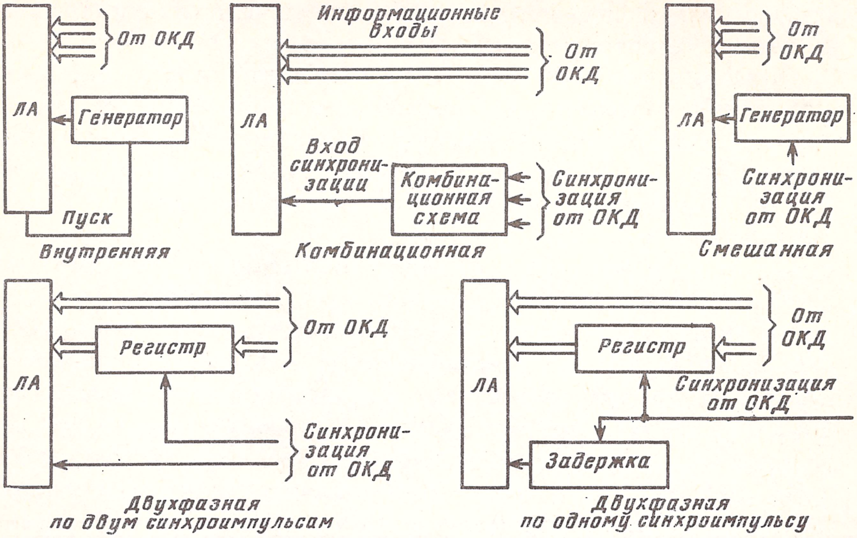


Рис. 7.6. Организация систем синхронизации ЛА

работы: "начало" – запуск осуществляется сразу после момента запуска; "сдвиг" – до и после момента запуска; "конец" – измерения до момента запуска; после задержки на *n* тактовых импульсов; после прихода двух запускающих слов; после прихода двух запускающих слов и задержки.

Таким образом, выбирая режимы анализатора, т.е. варьируя запускающие слова, тактовые импульсы (синхронизацию), цифровые задержки, выбирая точки подключения информационных каналов, можно проверять достаточно сложные цифровые устройства и системы.

Рассмотрим простейшие примеры использования ЛА [99–102].

Проверка содержимого постоянных запоминающих устройств (ПЗУ). Импульсы, тактирующие адресный счетчик, должны использоваться в качестве тактовых импульсов для записи информации в память анализатора. К информационным входам анализатора подключаются адресные входы ПЗУ и выход информации. Организуя запуск по адресным входам, начиная с адреса 0000..., на экране индикатора можно получить таблицу истинности ПЗУ по первым 16 ад-

ресам. Для дальнейшего просмотра необходимо использовать цифровую задержку.

Когда выходы ПЗУ работают на общую шину и съем информации идет по мере необходимости (по сигналу "выбор кристалла"), используется вход признаков. В этом случае в память анализатора будет заноситься только то, что выдает ПЗУ в общую шину.

Возможна и работа с запуском по входу информации. В этом случае определяется адрес по известному содержимому памяти.

Проверка оперативных запоминающих устройств (ОЗУ). Принципиально проверка содержимого ОЗУ не отличается от проверки содержимого ПЗУ. Используя входы признаков для подачи сигналов "Выбор кристалла" и "Запись – чтение", можно выбирать требуемую информацию от конкретной интегральной схемы памяти, установленной на плате с большим количеством ИС. Одновременно с этим указанные сигналы разрешают просматривать не только записанную информацию, но и контролировать содержимое информации, поступающей на запись.

Проверка работы линии коллективного пользования (ЛКП). При помощи ЛКП производится обмен информацией между отдельными приборами, входящими в информационно-измерительную систему. Обмен информацией сопровождается сигналами синхронизации, которые могут быть использованы в качестве тактовых импульсов для записи информации в память анализатора.

Возможны следующие случаи:

1) если в качестве тактового импульса используется перепад 1/0 сигнала СД ("Сопровождение данных"), то в память анализатора будет записываться информация, выдаваемая передатчиком в линию;

2) если используется перепад 1/0 сигнала ДП ("Данные приняты"), то в память анализатора будет записываться информация, принятая приемником.

Выбирая тот или иной режим работы анализатора ("начало" или "конец"), требуемое запускающее слово и цифровую задержку, наблюдают информацию, проходящую по линии.

При отыскании причин, приводящих к сбою, удобен режим "конец". Предположим, что на каком-то такте работы

по линии проходит заведомо ложная информация. Например, печатающее устройство печатает знак минус, когда известно, что измеряется положительное напряжение. В этом случае условия запуска устанавливаются по коду знака минус. На индикаторе будет зафиксирована ошибочная информация и информация о полученных ЛА данных за 15 тактов до момента сбоя. Анализ полученных данных укажет на причину сбоя.

Если использовать вход признаков, подключив к нему линию УП ("управление"), то возможна выборка всех универсальных и адресных команд. В качестве тактового импульса используется сигнал СД.

Если в качестве условия запуска использовать линию ЗО ("запрос на обслуживание"), то можно просмотреть весь процесс идентификации запроса на обслуживание, т.е. выяснить, какой из передатчиков выдал сигнал и каков его байт состояния.

При работе с ЛКП, которая работает в асинхронном режиме, удобны АВД, которые также работают в асинхронном режиме и позволяют поэтому контролировать весь процесс синхронизации.

7.3. ПРИМЕНЕНИЕ ЛОГИЧЕСКИХ АНАЛИЗАТОРОВ ДЛЯ ОТЛАДКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

По оценкам доля программного обеспечения в цифровой технике к концу десятилетия вырастет до 60 %, а в микропроцессорной технике до 70–90 %. Следовательно, основным путем уменьшения затрат и сокращения сроков проектирования цифровой аппаратуры является ускорение разработки программного обеспечения, в частности, отладки и испытаний программ.

Для успешного решения этих задач разработчику программного обеспечения необходимы количественные показатели работы программ, особенно при работе и отладке в режиме реального времени.

В настоящее время применяются два основных метода определения количественных показателей работы программ: метод вставок-ловушек, фиксирующий количество обращений к данному блоку, и метод с применением прог-

рамм контроля, определяющий время выполнения заданного модуля.

Обоим методам свойственны существенные недостатки. Методы не позволяют определить все необходимые количественные показатели работы программы.

Создание как вставок-ловушек, так и программ контроля неизменно увеличивает накладные расходы и сроки создания программного обеспечения, ухудшает работу программ в реальном масштабе времени. При ограниченной емкости памяти использование обоих методов часто становится невозможным.

Потребность в аппаратуре, позволяющей фиксировать и выдавать разработчику количественные показатели работы программ, привела к созданию специализированных приборов, получивших название программных анализаторов.

Программные анализаторы функционируют в нескольких режимах, в каждом из которых определяются количественные показатели работы программ. В режиме активности программ определяется либо число вхождений в конкретный блок (блоки) программной памяти, либо время, затрачиваемое на выполнение конкретного блока (блоков) программного кода. При этом учитываются все процессы, связанные с работой указанного кода в этой области, т.е. операции чтение, запись в память, чтение и запись в стек и т.д., причем каждое такое обращение считается одним вхождением для данной области.

На основе результатов измерения числа вхождений и времени выполнения отдельных модулей и программы в целом, можно определить наиболее активно работающие модули и части программы, время выполнения которых чрезмерно велико, что отрицательно сказывается на общей производительности системы. Данный режим позволяет определить процентный состав обращений, при которых время работы модуля является максимальным, средним и минимальным.

Режим измерения активности памяти аналогичен режиму измерения активности программ, отличается от последнего тем, что ориентирован на контроль использования памяти, а не активности работы программ.

При этом накапливаются данные по числу вхождений (событий, обращений) и по времени для указанных пользователем блоков памяти.

Данный режим снимает общую динамическую картину использования адресного пространства системы.

Режим измерения частоты использования модулей ориентирован на вычисление отрезков времени между последовательными обращениями к данному модулю. Таким образом, этот показатель дополняет режим измерения длительности выполнения модуля. Данный режим вычисляет среднее время обращения к указанному модулю, т.е. определяет количественный показатель "спроса" на модуль.

Режим измерения длительности интервалов между модулями служит для определения распределения временных интервалов между двумя кодовыми сегментами, будь то модули-подпрограммы, процедуры или другие части программы.

Чрезмерно большие межмодульные временные интервалы могут свидетельствовать о том, что кроме всего прочего имеются не предусмотренные разработчиком состояния ожидания, узкие места или "пробки" в аппаратуре либо увеличенные накладные затраты операционной системы.

Режим измерения межмодульного графика служит для подсчета числа прямых передач данных между любыми двумя программными модулями, причем он позволяет определить количественную оценку графика для некоторого множества выбранных пар модулей. Режим межмодульного графика позволяет установить причину снижения эффективности системы, устанавливая чрезмерную активность между одними модулями и низкую активность между другими, что может не соответствовать предполагаемому режиму работы программы. При отсутствии средств контроля, практически не вносящих накладных затрат, выявить подобные проблемы исключительно трудно.

Определение количественных показателей в процессе работы программы помогает выявить узкие места, что позволяет обоснованно вносить изменения в алгоритм и принимать рациональные компромиссные решения по распределению функций между аппаратными и программными средствами.

Примером программного анализатора служит отчечественный прибор ПАЛС, а также прибор 64310А, входящий в систему логического проектирования 64000 фирмы Hewlett Packard. Анализатор этого типа позволяет разработчикам

более эффективно выполнять почти все этапы цикла проектирования программного обеспечения. Их можно с успехом использовать для определения параметров, тестирования и оптимизации программ. Анализатор реализует все выше описанные режимы, имеет возможность легко переходить из одного режима работы в другой, что достигается при помощи программируемых "экранных" команд прямого синтаксиса.

При проведении измерений производится суммирование собираемых данных с учетом ранее собранных данных измерения и осуществляется их совместная математическая обработка. Важным достоинством анализатора является возможность управления оператором точностью измерения в пределах от 51 до 99 %. Результаты вычислений представляются на экране в виде гистограмм или непосредственно в табличном виде.

Представление результатов измерения производится без прерывания процесса сбора данных измерения. Такая одновременность позволяет разработчику выбирать наиболее информативный режим воспроизведения данных измерения для конкретной задачи, сопоставлять различные виды представления и запоминать необходимые сведения, не нарушая процесса измерения. Подобные возможности не только экономят время, но и способствуют проведению экспериментов, позволяют ставить различные вопросы относительно работы программы и получать на них конкретные ответы.

Для иллюстрации манипуляций, проводимых оператором с помощью ЛА, рассмотрим пример организации просмотра программы в МПС.

Информационные входы анализатора подключаются к адресной шине и шине данных. Синхронизация осуществляется от центрального процессора, используются дополнительно входы признаков П1, П2 (рис. 7.7).

Если в качестве слова запуска установить адрес ячейки начала программы, а на вход признаков подать синхроимпульс машинного цикла, то на устройстве отображения наблюдается вся программа МПС. Для просмотра только кодов команд, содержащихся в ПЗУ, дополнительно на один из входов признаков надо подать синхроимпульс цикла М1 – машинного цикла чтения команды (для микропроцессора К580ИК80А, который формируется на шине данных в раз-

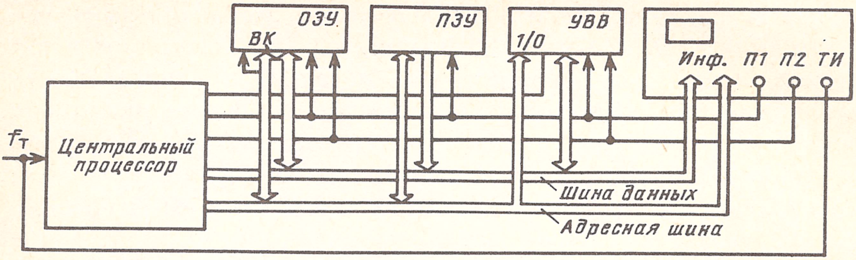


Рис. 7.7. Просмотр программы МПС

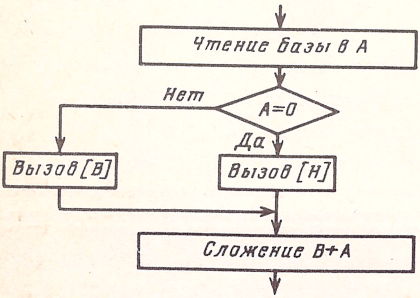


Рис. 7.8. Пример отладки ветвящихся программ

ряде Д5). Для просмотра подпрограмм запускающим словом служит адрес ячейки, в которой записана команда CALL. При необходимости просмотра информации после нескольких петель подпрограмм применяют задержку запуска.

Как отмечалось выше, для анализа программ необходимо использовать более сложные виды запуска. Например, для проверки ветвей программы между блоками *A* и *C* (рис. 7.8) необходимо использовать последовательный запуск по совпадению нескольких параллельных слов. Так, имеются два пути *ABC* и *AHC*, они различаются путем использования запуска по словам *AB* и *AH* (адреса команд вызова *B* и вызова *H* соответственно).

7.4. ПОСЛЕДОВАТЕЛЬНЫЕ АЛГОРИТМЫ КОМПЛЕКСНОЙ ОТЛАДКИ МПС

Микропроцессорная система может быть описана на различных уровнях представления. Например, в [105] рассмотрены следующие уровни представления МПС: "черный ящик", структурный, программный (языковый, команд процессора), логический (регистровых пересылок, переключа-

тельных схем), схемный. При разработке системы происходит переход от одного уровня ее представления к другому, более детальному.

При отладке вычислительного процесса, происходящего в системе по факту проявления ошибки на самом верхнем, функциональном уровне, необходимо найти место ее возникновения, постепенно переходя с одного уровня на другой. Можно выделить следующие уровни возникновения и поиска ошибок:

функциональный (при поиске ошибок это уровень контроля);

программно-временной;

программно-структурный;

программно-информационный;

аппаратный уровень логических сигналов;

аппаратный уровень электрических сигналов.

Ошибка, возникающая на нижнем уровне, может проявиться на одном или нескольких более высоких уровнях. Каждому из уровней возникновения ошибки соответствует определенный этап проверки системы. На каждом из уровней вычислительный процесс может быть представлен соответствующей моделью.

Структурно-временная модель M_1 описывает последовательность хода вычислительного процесса. Она также отражает временные процессы, происходящие в системе. Она представляет собой ориентированный граф $G_1(V^1S^1)$, где вершинам v соответствуют операторы управляющей программы, а дугам s – управляющие связи между операторами. Каждой вершине графа поставлена в соответствие величина τ, которая представляет собой время выполнения соответствующего вершине оператора.

Информационная модель M_2 отражает ход вычислений или обработки данных. Она представляет собой ориентированный граф $G_2(V^2S^2)$, где вершинам соответствуют операторы программы, дугам – информационные связи. Дуга s_{ij} существует в том случае, если результат выполнения v_i является исходной информацией для работы v_j. Исходные данные для работы программы в целом представлены вершинами, принадлежащими подмножеству $V_н$.

Модель M_3 представляет собой формализованное описание взаимосвязей между операторами программы и аппаратными компонентами МПС. Это двудольный граф G_3,

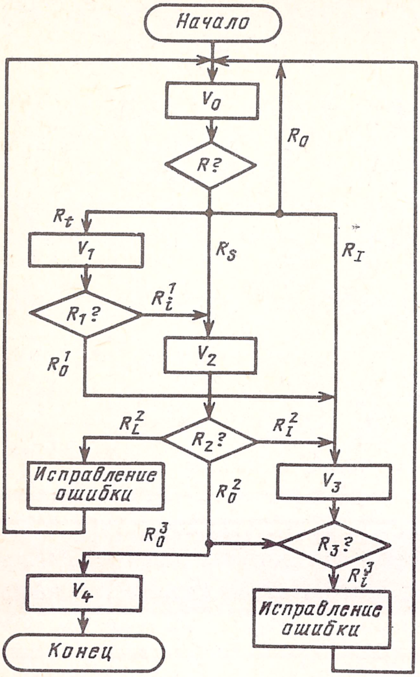


Рис. 7.9. Общий алгоритм хода отладки

множество вершин V_1 которого представляют собой операторы программы, а множество вершин V_2 – аппаратные компоненты МПС. Дуга s_{ij} между v_i и v_j существует в том случае, если компонент, соответствующий v_j, участвует в выполнении оператора, соответствующего v_i. Например, в выполнении команды MOV A, M МП К580ИК80 участвуют следующие аппаратные компоненты: ПЗУ, ОЗУ, магистраль передачи данных, микропроцессор.

Таким образом, совокупность частных моделей образует общую многоуровневую модель, отражающую выполнение системой программы с различных точек зрения.

Общий алгоритм отладки. При выполнении комплексный отладки поиск ошибок представляет собой итерационный процесс, при котором после локализации ошибки следует ее исправление, а затем продолжение проверок. Блок-схема алгоритма показана на рис. 7.9. Каждый блок соответствует этапу выявления ошибки на определенном уровне ее возникновения или проявления и характеризуется информационной функцией:

$$\Phi = \langle S, E, R, T, L, C \rangle,$$

где S – вектор входных тестовых сигналов; E – система соответствующих эталонов; R – вектор возможных реакций; T – вектор трассы, т.е. совокупность промежуточных результатов выполнения ВП; L – вектор точек наблюдения, т.е. снятия реакций; C – вектор синхронизации, характеризующий моменты времени снятия реакции.

Блок V_0 представляет собой этап контроля МПС на программном уровне. Точки подачи тестов и съема реакций – это точки в программе, которые должны быть определены заранее, так же как и векторы S и E. На этапе контроля не рассматриваются векторы трассы T и векторы синхронизации C. Вектор реакций системы состоит из следующих составляющих: R_0 – реакция, совпадающая с эталоном; R_s – реакция, не совпадающая с эталоном по структуре программы; R_t – реакция, не совпадающая с эталоном по времени; R_I – реакция, не совпадающая с эталоном по вычисленным в программе данным.

Возможно, что реакция не совпадает с эталоном сразу по двум или по всем трем параметрам. В этом случае существует система приоритетов. Высшим приоритетом обладает реакция R_s, затем R_t и, наконец, R_I, т.е. ошибку по структуре, времени и данным рассматриваем как структурную и переходим в этом случае к поиску ее на структурном уровне. Ошибку по времени и данным считаем временно́й. Особо следует выделить реакцию зацикливания $R_{t=\infty}$. Ее можно считать и структурной, и временно́й одновременно. Она обладает высшим приоритетом, как R_s.

В результате работы каждого блока могут быть приняты следующие решения: 1) полное окончание всех проверок; 2) окончание проверки данного пути вычислительного процесса; 3) повторение данной проверки на данном уровне с изменением информационных характеристик; 4) повторение данной проверки с переходом к следующему уровню поиска ошибки.

На этапе контроля, получив одну из ошибочных реакций, еще нельзя сказать ничего определенного о характере ошибки и ее местоположении. Детализация поиска ошибки происходит путем перехода к блокам V_1, V_2, V_3 алгоритма.

Блок V_1 соответствует программно-временно́му уровню отладки. На этом этапе возможно возникновение двух групп реакций: R^1_0 – совпадение всех реальных t_{ir} с эталонными t_i и R^1_i – несовпадение t_{ir} с t_i. Если на этапе контроля совпадение реакции с эталоном говорит об отсутствии ошибки, то на всех других уровнях это означает невозможность обнаружения существующей ошибки и необходимость перехода на следующий уровень. Поэтому при возникновении реак-

ции R_0^1 осуществляется переход типа 3 к структурному уровню отладки (блок V_2). При возникновении реакции R_i^1 осуществляется переход типа 3 к блоку V_3 для более детального рассмотрения ошибочного цикла. Таким образом, на данном уровне ошибка может быть локализована с точностью до цикла программы, исправление ошибки здесь не предусмотрено.

В блоке V_2 выполняется поиск неверного перехода в ходе вычислительного процесса. Точность обнаружения ошибки в этом случае – до команды ассемблера. Неверная реакция R_i^2 может быть отнесена к следующим группам:

R_{iL}^2 – логическая ошибка в программе. Например, вместо перехода JZ записан переход JNZ. Такая ошибка может быть исправлена на данном уровне и осуществлен в алгоритме переход типа 2;

R_{iI}^2 – информационная ошибка в программе. Неверно условие перехода. Такую ошибку необходимо локализовать на информационном уровне (переход к блоку V_3);

R_0^2 – ошибка, которая заключается в том, что неверный переход происходит после линейной команды (не команды ветвления) либо после правильной команды ветвления с правильным условием перехода. В этом случае необходимо детальное исследование данной команды и предшествующей ей на аппаратном уровне (переход к блоку V_4).

Блок V_3 ориентирован на выявление ошибок в информационном графе программы. Неверные реакции могут быть следующими:

R_i^3 – ошибка, заключающаяся в неверной записи операндов или оператора, переполнение разрядной сетки и т.д., т.е. такая ошибка, что неверный результат r_i в вершине v_i информационного графа $G_2(V^2 S^2)$ есть следствие программной ошибки в одной или нескольких вершинах, предшествующих v_i и смежных с ней. Эта ошибка может быть исправлена на должном уровне и осуществлен переход типа 2;

R_0^3 – ошибка, возникшая следующим образом: результат r_i вершины v_i графа $G_I(v_I s_I)$ неверен, хотя результаты всех вершин, предшествующих v_i и смежных с ней, верны. В этом случае необходимо исследование команды на аппаратном уровне (переход к блоку V_4).

Блок аппаратной отладки V_4 определяет местоположение неисправности в аппаратных компонентах с точностью до микросхемы.

Таким образом, для проведения комплексной отладки по данному алгоритму необходимо определить заранее функцию Ф для каждого блока и разработать алгоритм локализации ошибок внутри каждого блока. Принцип локализации ошибок очевиден и одинаков для всех блоков программных уровней – это сравнение реальной и эталонной трасс вычислительного процесса. Под трассой здесь понимается совокупность промежуточных точек, через которые проходит вычислительный процесс. Точка, в которой вычислительный процесс отклоняется от эталонного, фиксируется, и ошибка локализуется в ней. Рассмотрим теперь подробнее каждый блок алгоритма и найдем некоторые из информационных характеристик.

Этапы отладки программного обеспечения МПС. Предлагаемая методика предусматривает комплексную отладку системы в динамике, т.е. при подключении, взаимодействии и взаимном влиянии всех подпрограмм и внешних устройств или их имитаторов. Ход вычислительного процесса должен быть непрерывным.

При фиксации временно́й ошибки предпринимаются попытки локализовать ее на этапе проверки циклов управляющей программы. Происходит регистрация точек ВП, являющихся входами в циклы на данном участке каждый раз при их появлении. Совокупность таких точек образует реальную трассу ВП: $T = (k^1_1\ k^1_2\ \dots\ k^1_{n1},\ k^2_1\ \dots\ k^2_{n2},\ k^m_1\ \dots\ k^m_{n_m})$, где n_i – число появлений i-точки, m – количество циклов на участке. Системой эталонов при этом будет служить множество рассчитанных при составлении тестов контроля величин $n_{i\text{эт}}$, которые представляют собой число тактов до завершения i-го цикла при данной реализации ВП. При первом же несовпадении n_i и $n_{i\text{эт}}$ ошибка локализуется в i-м цикле. При этом прибор, осуществляющий наблюдение, должен быть подключен к адресной шине МП отлаживаемой системы, так как все точки k реально представляют собой адреса операторов-входов в циклы, т.е. $l_i \in L$ – выводы адресной шины, их количество равно разрядности адресной шины.

Следующим является этап проверки структуры вычислительного процесса. Участок ВП, начавшийся в точке X,

должен завершиться в точке Y. Это и есть на данном этапе входо-выходной тест вычислительного процесса. Критерием трестирования, очевидно, является прохождение всех путей алгоритма. Работа данного блока заключается в том, что при нарушении хода ВП должен быть найден адрес команды, после которой произошел неправильный переход. На данном уровне проявляются информационные ошибки и ошибки в узлах аппаратуры.

Локализация ошибочной команды происходит путем сравнения эталонной адресной трассы и реальной. Эталонной адресной трассой считаем последовательность адресов проверяемого участка программ, полученную в результате трансляции программы: $E = (A_{E1}, A_{E2} \ldots A_{En_1})$. Реальная трасса – последовательность реальных адресов: $T = (A_{R1}, A_{R2}, \ldots, A_{Rn_2})$. При правильной структуре ВП все $A_{Ri} = A_{Ei}$.

Для сокращения времени отладки в качестве элементов системы эталонов можно выбрать не все адреса данного пути программы, а лишь адрес условных и безусловных переходов. Реальные переходы ВП образуют соответствующую трассу $T_р$. При первом же несовпадении A_{Ei} и A_{Ri} ошибка локализуется в меньшем по значению адресе, т.е. если произошел "лишний" переход, которого нет в эталонной трассе, то ошибка локализуется в команде с адресом, соответствующим "лишнему" переходу; если же не произошло необходимого перехода, то ошибка локализуется в команде с адресом, соответствующим необходимому переходу.

Физическими точками наблюдения $l_i \in L$ на данном этапе являются те точки системы, наблюдая которые можно получить адресную трассу $T_а$ вычислительного процесса. Очевидно, эти точки – выводы адресной шины микропроцессора, их количество – разрядности адресной шины. Что касается моментов времени, в которые происходит фиксация адресов, то это должны быть моменты установившегося значения адресов на адресной шине. Это значит, что они будут совпадать с моментами времени синхросигналов микропроцессора.

На этапе отладки информационной части вычислительного процесса в качестве входо-выходной тестовой последовательности используется информационная часть тестово-

го набора, полученного для этапа контроля МПС. Работа блока V_3 заключается в поиске команды, начиная с которой ВП начал вычислять неверные результаты. И в этом случае локализация ошибки основана на сравнении эталонной и реальной трасс, однако существуют значительные трудности в определении эталонной трассы информационного этапа. Построить трассу E в данном случае – означает найти все промежуточные результаты работы ВП после выполнения каждого оператора. Поэтому алгоритм локализации ошибки в информационном графе составлен так, чтобы вычислить не все, а только некоторые промежуточные результаты того пути, на котором произошла информационная ошибка.

После завершения отладки, когда в ответ на входной набор S получаем реакцию R, совпадающую с эталоном E, соответствующую реальную трассу $T_р$ можно считать эталонной, заполнить и использовать в дальнейшем при диагностировании данной МПС на этапе ее эксплуатации.

Точками наблюдения l_i являются выводы шины данных, по которой передаются из памяти в МП команды программы. Точки синхронизации совпадают с собственной синхронизацией МП.

Функциональная отладка узлов аппаратуры МПС. При автономной отладке аппаратных компонентов МПС используются специальные тестовые программы, стремящиеся охватить проверкой все возможные режимы работы и все структурные составляющие контролируемого узла. При функциональной комплексной отладке разработчик учитывает, что МПС обычно выполняет определенную функцию согласно программе, записанной в ПЗУ. Эта функция может использовать возможности системы частично. При функциональном подходе к отладке проверкой охвачены лишь те части и режимы работы узла, которые участвуют в выполнении этой функции, определяемой прикладной программой.

Модель М4 определяет связь программных операторов и аппаратных компонентов (АК) МПС. Ошибка вычислительного процесса, проявившаяся при проходе определенного пути программы, ставит под сомнение исправность всех узлов аппаратуры, участвующих в вычислительном процессе на этом пути, т.е. подозреваемыми являются все вершины

$v_{ii} \in v_1^j$ графа G_A, где j – номер пути программы. Предполагаемая методика позволяет сократить число подозреваемых узлов, подлежащих более детальной проверке.

Каждый переход к более низкому уровню отладки локализует ошибку с определенной степенью точности и сокращает подозреваемый участок ВП. В момент перехода к аппаратному уровню отладки этот участок сокращен до одного конкретного оператора программы. Поэтому необходимо рассмотреть подграф G_x графа G_A; подмножество вершин V_i^x данного подграфа будет состоять из единственной вершины V_x, соответствующей оператору, в котором зафиксирована ошибка. Подмножество вершин V_1^x состоит из вершин, смежных с v_x.

Аппаратный состав МПС может быть весьма разнообразен, однако некоторые компоненты обязательно входят в состав всех МПС. Поэтому множество V_1 графа G_A будут составлять следующие элементы:

блоки микропроцессора: аккумулятор, счетчик команд, регистр команд, дешифратор команд, регистры общего назначения, арифметическое устройство, схемы синхронизации и управления, регистр состояния МП, канал прямого доступа к памяти;

прочие узлы МПС: совокупности ячеек ПЗУ, ячеек ОЗУ, внешних устройств, устройств сопряжения, вспомогательных схем (буферные ЗУ, схемы прерывания, генератор токовых сигналов и т.д.).

Дальнейшее сокращение подмножества подозреваемых узлов связано с анализом реакции R на предыдущем этапе. Как видно из алгоритма на рис. 7.9, ошибка, возникшая в АК, может проявиться в структуре либо в информационной части программы. Реакция типа R_0^2, возникшая в результате выполнения оператора V_x, очевидно, связана с фазой выборки команды V_x. Список АК, участвующих в фазе выборки, может быть определен однозначно и представляет собой подмножество $V_x^{\text{выб}}$: ⟨счетчик команд, ячейка ПЗУ, регистр команд, дешифратор команд, регистр состояния МП, устройства сопряжения МП⟩.

Реакция типа R_0^3 указывает на то, что ошибка возникла в фазе исполнения оператора V_x. Может быть для данной

МПС однозначно определено подмножество $V_x^{\text{исп}}$, элементами которого являются АК, участвующие в фазе исполнения команд: $V_x^{\text{исп}}$: ⟨ячейка ПЗУ, дешифратор команд, регистр команд, РОН, АЛУ, канал ПДП, ячейка ОЗУ, ВУ и т.д.⟩.

Множество V_x подозреваемых АК будет получено следующим образом:

$$v_x := V_x \cap V_x^{\text{выб}} \text{ при } R = R_0^2;$$

$$V_x := V_x \cap V_x^{\text{исп}} \text{ при } R = R_0^3.$$

Каждый аппаратный компонент характеризуется списком точек наблюдения, непосредственно доступных разработчику. Среди таких точек можно выделить точки, соответствующие каналам управления адреса и данных:

$L_i^u : \langle l_{i1}^u, l_{i2}^u, ..., l_{in}^u \rangle$ – список точек наблюдения шины управления разрядностью n i-го узла;

$L_i^A : \langle l_{i1}^A, l_{i2}^A, ..., l_{im}^A \rangle$ – список точек наблюдения шины адреса разрядностью m;

$L_i^D : \langle l_{i1}^D, l_{i2}^D, ..., l_{ik}^D \rangle$ – список точек наблюдения шины данных разрядностью k.

Объединив все списки точек наблюдения L_i, соответствующие всем подозреваемым узлам v_i, получим список L необходимых точек наблюдения на данном этапе отладки.

Для проведения отладки должны быть определены входные тестовые сигналы S, а также эталоны E для каждой точки наблюдения объекта. Как уже было отмечено, точки управления и входные тестовые сигналы являются общими для всех этапов отладки, а эталоны должны быть определены для каждого этапа в отдельности.

Иерархическая структура процесса отладки предполагает на нижних уровнях отладки, что ошибки, возникающие на более высоких уровнях, выявлены и исправлены. Поэтому при поиске выходных тестовых данных для подозреваемого оператора и связанных с ним АК будем считать эталоном запись этого оператора в машинных кодах в программе, которая состоит из следующих полей: p_1 – адрес команды

в ПЗУ, p_2 – код операции, p_3 – информация об операндах. Запись команды содержит достаточно информации для того, чтобы (без подачи дополнительных входных тестовых сигналов) предположить, каких результатов следует ожидать в соответствующих точках наблюдения (почти во всех). Поэтому возможно построение тестовой таблицы со следующей структурой:

Код команды	Все возможные списки точек наблюдения

Строки таблицы соответствуют командам микропроцессора, столбцы – всем возможным спискам точек наблюдения системы. На пересечении строк и столбцов для каждого списка точек наблюдения имеем список эталонных сигналов.

Например, рассмотрим команду MOV A, M и связанную с ней АК оперативной памяти. Соответствующие списки L для данной АК следующие:

$$L^{ч}_{ОЗУ} : \langle l_1^u \, l_2^u \rangle,$$

где l_1^u – сигнал чтения из памяти; l_2^u – сигнал записи в память;

$$l^A_{ОЗУ} : \langle l_1^A \ldots l_{16}^A \rangle; \quad l^D_{ОЗУ} : \langle l_1^D \ldots l_8^D \rangle.$$

Соответствующие списки эталонов:

$$E^u_{ОЗУ} : \langle 1, 0 \rangle;$$

$$E^A_{ОЗУ} : \langle \text{двоичный код } p_3 \rangle;$$

$E^D_{ОЗУ} : \langle x \rangle$ – это значит, что для команды шина данных ОЗУ не подлежит проверке, так как была проверена выше при поступлении этих данных в ОЗУ. Операнд, являющийся источником, и соответствующие ему списки АК не нуждаются в проверке, так как предполагается, что они были проверены ранее.

Данная таблица должна включать в себя все команды микропроцессора, на котором построена отлаживаемая МПС. Очевидно, этап аппаратной отладки ориентирован на отладку МПС, построенных на микропроцессорах одного ти-

па. Вероятно, таблицу тестовых эталонов можно сделать универсальной, так как команды различных МП весьма близки и по смыслу и по структуре.

Таким образом, система E эталонов выбирается из таблицы, при этом используется известная запись оператора и определенный ранее список L необходимых точек наблюдения.

Для ряда команд МП, связанных с арифметико-логическим устройством (команды типа ADD, SUB, AND и т.д.), невозможно заранее указать результат, содержащийся в приемнике после выполнения команды. В этом случае необходимо либо вмешательство оператора в работу автоматизированной системы отладки и введение рассчитанного вручную эталона, либо наличие в системе отладки статической проверки данных функций АЛУ с подачей входных тестовых сигналов для выполнения только одной данной команды.

При динамической проверке узлов МПС следует анализировать временные характеристики логических сигналов на выводах аппаратных компонентов. В систему E эталонов должно входить время τ, равное фронту логического сигнала при переключении данного разряда.

Глава 8

ВНУТРИСХЕМНЫЕ ЭМУЛЯТОРЫ КАК СРЕДСТВО НАЛАДКИ микроУВК

8.1. ОБЩИЕ СВЕДЕНИЯ

Одним из значений английского слова emulation является русское слово "подражание". Отсюда следует, что эмуляция – это подражание одной ЭВМ с помощью другой. В больших ЭВМ эмуляция осуществлялась путем создания модели эмулируемой ЭВМ на программном или микропрограммном уровне. В результате получали возможность запуска без всяких изменений программ эмулируемой ЭВМ на эмулирующей, даже если они были написаны в машинном коде эмулируемой ЭВМ.

Внутрисхемная эмуляция – это порождение микропроцессорной техники, которое заключается в том, что эмуляция разрабатываемой или диагностируемой микроЭВМ осущест-

вляется на аппаратном уровне. Это обеспечивается за счет особенностей архитектуры отладочного средства, именуемого внутрисхемным эмулятором (ВСЭ). В результате пользователю предоставляется возможность одновременного использования внутренних аппаратных ресурсов как ВСЭ, так и эмулируемой микроЭВМ. Характерной чертой внутрисхемной эмуляции является то, что пользователь по своему желанию может выбирать необходимые ему ресурсы эмулируемой микроЭВМ как из состава этой микроЭВМ, так и из аппаратуры ВСЭ. В результате образуется удивительный по своим возможностям симбиоз отлаживаемой микроЭВМ и отладочного средства, обладающего мощной операционной системой, широким спектром устройств ввода-вывода, в том числе и дисковыми ЗУ. Пионером в создании ВСЭ явилась фирма Intel (США) со своим Интеллеком. В настоящее время различными фирмами разработан широкий спектр ВСЭ для различных типов микропроцессоров. Так, только фирма Intel уже выпускает ВСЭ Интеллек серии IV, который позволяет эмулировать микроЭВМ, построенные на базе различных микропроцессоров и однокристальных микроЭВМ, разработанных в этой фирме. В нашей стране разработаны ВСЭ для микропроцессоров К580ИК80, например КРАМ [110], микропроцессоров К1810 ВМ86 – КРАМ 2 и СМ1810 [20].

Управление работой ВСЭ осуществляется с помощью операционной системы (ОС) типа ISIS. Такая операционная система состоит из двух частей: монитора, который размещается в ПЗУ ВСЭ, и непосредственно дисковой операционной системы, выполняющей большинство функций управления.

Монитор осуществляет проверку наличия ОС на дисководе, его ввод в ОЗУ ВСЭ с диска и передачу управления введенной программе. В том случае, когда ОС на дисководе отсутствует, ВСЭ остается под управлением программы "Монитор", которая обеспечивает выполнение ряда обычных для таких программ функций, таких, как обеспечение диалогового режима работы с клавиатурой и дисплеем ВСЭ, загрузка и запуск программы пользователя из памяти ВСЭ, ее останов, пересылка массивов информации из одной области памяти в другую, заполнение ОЗУ ВСЭ константой, заданной с клавиатуры, изменение содержимого памяти и т.д. Программа "Монитор" располагается обычно в той области памяти, которая содержит ячейку памяти, адресуе-

мую микропроцессором, на базе которого выполнено ЦПУ ВСЭ, после подачи на него сигнала "Сброс".

Дисковая часть ОС состоит из управляющей программы, ввод которой в память ВСЭ осуществляется после включения ВСЭ, и некоторого множества файлов, которые обеспечивают весь сервисный набор функций ВСЭ. Управляющая программа предназначена для ввода требуемых файлов программы с системного диска (диск, на котором расположена ОС) и рабочих дисков пользователя и позволяет выполнить некоторый набор команд, а именно:

непосредственное выполнение программ. Это команда, позволяющая передать управление программе, имя которой задано с клавиатуры;

выполнение программы под управлением "Монитора". При использовании этой команды передача управления "Монитору" производится после ввода программы пользователя с диска. После передачи управления оператор работает с клавиатурой по правилам работы с "Монитором";

выполнение программы без использования клавиатуры. Команда позволяет осуществить работу ВСЭ под управлением некоторого управляющего файла, в котором оператор формирует все управляющие операторы, необходимые ему при запуске его программы;

преобразование кодов. Это группа команд, которые позволяют преобразовывать информацию из одной системы счисления в другую;

команды редактирования текста. Это группа команд, которые позволяют осуществлять редактирование текстов путем использования управляющих и информационных клавиш клавиатуры ВСЭ;

команда введения текста в буфер. Позволяет ввести необходимый пользователю текст в ОЗУ ВСЭ;

команда печати текста;

команды поиска и редактирования цепочки текста в памяти ВСЭ;

запись текста в дисковый файл;

распечатка директории диска. Команда позволяет распечатать названия файлов, содержащихся на диске;

копирование дисковых файлов;

стирание дисковых файлов;

инициализация нового диска;

изменение имени дискового файла;

изменение описателей дискового файла;

команда сборки модульных программ. Осуществляет сборку программы, состоящей из нескольких отдельно написанных подпрограмм (модулей);

команда определения расположения. Производит привязку программы к заданным абсолютным адресам;

команда связи. Позволяет работать с библиотекой модулей, используемых при создании программы пользователя.

Набор команд, которые могут выполняться управляющей программой дисковой ОС ВСЭ, зависит от конкретного типа ВСЭ. Выше была перечислена часть команд, которая содержится в управляющей части ОС ISIS.

Наиболее существенное расширение функций, которое представляется пользователю ВСЭ, обеспечивается за счет дополнительных управляющих программ, которые являются самостоятельными файлами, хранящимися на системном диске. Эти файлы, как правило, содержат программы:

подготовки и редактирования текста;

управления процессом внутрисхемной эмуляции;

трансляторы с языков Ассемблера для допускаемых в данном ВСЭ микропроцессоров;

компиляторы с языков высокого уровня, набор которых зависит от данного ВСЭ;

интерпретаторы языков высокого уровня, набор которых также зависит от ВСЭ.

При необходимости пользователь может расширять список таких системных файлов, включая в него написанные им самим или заимствованные управляющие программы или трансляторы и компиляторы.

В ВСЭ наиболее часто применяются следующие языки высокого уровня, адаптированные к конкретным микропроцессорам: ПЛ, Фортран, Бейсик, Паскаль.

Из описанного выше следует, что ВСЭ являются достаточно мощными вычислительными средствами, позволяющими разрабатывать программы на различных языках (машинных, Ассемблерах, языках высокого уровня) и производить их отладку в режиме реального времени, которая обеспечивается за счет внутрисхемного эмулятора.

Особенностью внутрисхемных эмуляторов является то, что в их состав обязательно должен входить микропроцес-

сор, используемый в эмулируемой микроЭВМ, или схема, имитирующая его работу. Эмуляция секционированных микропроцессоров, управляемых микропрограммами, осуществляется путем замены БИС, выполняющей функции блока микропрограммного управления, на специализированный блок внутрисхемной эмуляции. При работе такого ВСЭ пользователю предоставляется возможность использовать микрокоманды из памяти, эмулируемой микроЭВМ, или хранить их в памяти ВСЭ. Эти возможности позволяют производить не только отладку программ для микроЭВМ, но и отладку системы микропрограмм, а также наладку таких микроЭВМ.

8.2. ОПИСАНИЕ БЛОК-СХЕМЫ ВНУТРИСХЕМНОГО ЭМУЛЯТОРА

Современный ВСЭ, как правило, представляет собой микропроцессорную микроЭВМ с дисковыми ЗУ, в качестве которых используются накопители на гибких и твердых магнитных дисках преимущественно типа "Винчестер" с терминалом, состоящим из дисплея и клавиатуры и позволяющим осуществить диалоговый режим работы принтером, программатором, который используется для программирования БИС ПЗУ различных типов, и с другим периферийным оборудованием (например, устройством ввода или вывода с перфоленты и т.д.), которое может подключаться к ВСЭ для реализации необходимых пользователю функций. Укрупненная схема ВСЭ представлена на рис. 8.1.

На данной схеме представлена ВСЭ, имеющая в своем составе три отдельные микропроцессорные системы: центральную микропроцессорную систему, обладающую собственной постоянной и оперативной памятью, системой прерывания и средствами ввода-вывода, вспомогательную микропроцессорную систему ввода-вывода со своей памятью. К этой системе подключаются различные устройства ввода-вывода, в том числе и средства организации диалогового режима работы, блок внутрисхемной эмуляции с собственным микропроцессором и памятью, дисковый контроллер и общее для всех этих систем дополнительное ОЗУ.

Центральная микропроцессорная система работает под управлением программы "Монитор" или дисковой операционной системы типа ISIS. Она осуществляет управление работой остальных микропроцессорных ВСЭ и дискового

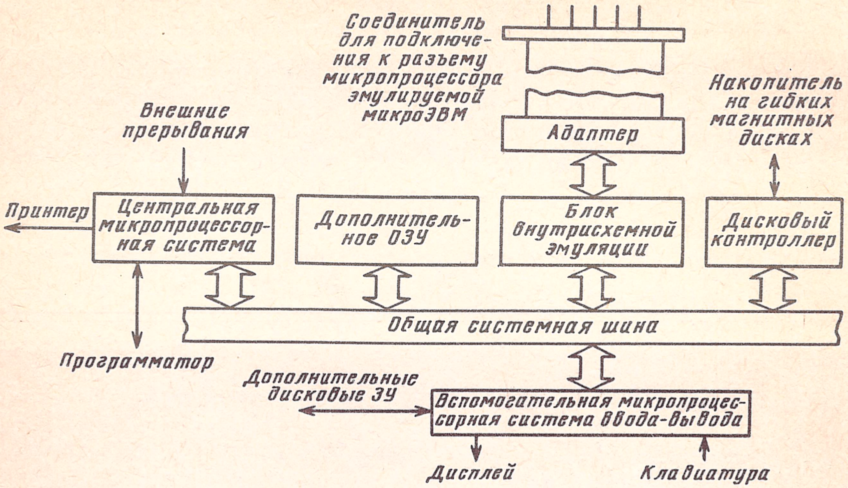


Рис. 8.1. Схема ВСЭ

контроллера. Однако остальные микропроцессорные системы, получив необходимую для инициализации своей работы управляющую информацию из центральной процессорной системы, дальнейшую работу осуществляют самостоятельно. После окончания работы эти микропроцессорные системы передают результаты и признаки окончания в центральную процессорную систему.

Как видно из рис. 8.1, все части ВСЭ являются обычными для любой микроЭВМ устройствами, за исключением блока внутрисхемной эмуляции. Этот блок должен заменяться в зависимости от типа микропроцессора, применяемого в эмулируемой микроЭВМ. Блок внутрисхемной эмуляции подсоединяется через специальный соединитель к процессорному устройству эмулируемой микроЭВМ, где этот соединитель устанавливается в соединитель микропроцессора на его место. Таким образом, определяется первое требование к проверяемой микроЭВМ: ее микропроцессор должен быть обязательно установлен на специальный соединитель и должен выниматься из него перед подключением к ВСЭ. Укрупненная схема блока внутрисхемной эмуляции представлена на рис. 8.2. Схемы на рис. 8.1, 8.2 позволяют реализовать следующие типичные для ВСЭ функции:

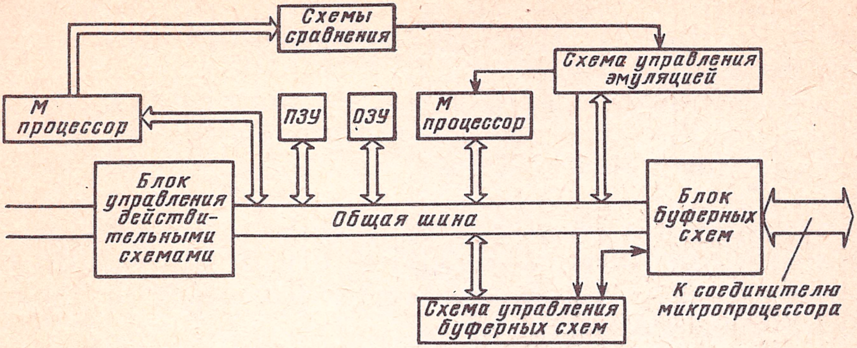


Рис. 8.2. Схема блока внутрисхемной эмуляции

программно-управляемое распределение массивов памяти;

программно-управляемое распределение портов ввода-вывода;

запоминание и вывод на устройство отображения содержимого внутренних регистров микропроцессора и состояние ряда его специальных выводов;

запоминание и вывод на устройство отображения некоторого количества последних выполненных микропроцессором машинных циклов с индикацией значений на шине адреса, содержимого счетчика команд, регистра команд и значений на шине данных микропроцессора;

выполнение программы пользователя в обычном или пошаговом режиме с возможностью останова по заданным оператором условиям или с автоматическим продвижением с выводом на дисплей информации о пройденной точке останова;

модификацию содержимого внутренних регистров микропроцессора, памяти, портов ввода-вывода как до начала выполнения программы пользователя, так и после ее промежуточных остановов по заданным условиям;

подсчет и отображение количества машинных тактов, выполненных в процессе исполнения эмулируемой программы, что позволяет оценить длительность работы этой программы;

останов программы пользователя в произвольный момент времени путем возбуждения запроса на прерывание от кнопок прерывания;

отображение информации в выбранной оператором системе счисления: двоичной, восьмеричной, десятичной или шестнадцатеричной;

задание точек останова в действительных адресах программы или в виде меток, используемых в языке Ассемблера или языке высокого уровня, на котором написана программа;

задание условия останова: по достижению адреса команды, указанного для останова, по чтению или записи в указанный адрес памяти или порт ввода-вывода, по записи или считыванию из стека;

считывание или запись информации на накопитель на гибких магнитных дисках, причем расположение адресуемой при этой операции памяти в эмулируемой микроЭВМ или в самом ВСЭ не имеет значения.

Управление работой ВСЭ и эмулируемой микроЭВМ осуществляется специальной управляющей программой типа ICE, которая позволяет реализовать кроме обеспечения функций, описанных выше, еще ряд дополнительных программно-сервисных функций. К таким функциям относятся: заполнение массива памяти некоторой заданной оператором информацией, отображение содержимого массива памяти на дисплее, изменение содержимого ячейки памяти или порта ввода-вывода с контролем записываемой информации, пересылка массивов памяти из одной области в другую с аналогичным контролем, отображение установленного распределения памяти, портов ввода-вывода, анализ и вывод на дисплей сообщений об ошибках, возникающих при эмуляции, и другие функции, набор которых может изменяться в зависимости от типа применяемого ВСЭ.

ВСЭ позволяют осуществлять диагностирование аппаратуры микроЭВМ, отладку разрабатываемых программ на разных стадиях разработки аппаратных средств микроЭВМ, поэтапную проверку правильности решений, принятых при разработке аппаратных средств. Отладка программ с помощью ВСЭ может производиться и при отсутствии разработанной аппаратной части микроЭВМ. При работе с ВСЭ пользователю предоставляются возможности использования Ассемблеров, языков высокого уровня, таких, как ПЛМ, ПЛМ-86, Бейсик, Фортран и др., использование редакторов текста и т.д.

8.3. ПРИМЕНЕНИЕ ВСЭ ДЛЯ ДИАГНОСТИРОВАНИЯ микроЭВМ

Диагностирование микроЭВМ с помощью ВСЭ начинается с изъятия микропроцессора микроЭВМ и установки на его место соединителя ВСЭ. После этого включаются блоки питания ВСЭ и эмулируемой микроЭВМ, запускается программа управления внутрисхемной эмуляцией. ВСЭ производит анализ наличия уровней питания и синхронизирующих сигналов в эмулируемой микроЭВМ и в случае их отсутствия выдает специальное диагностическое сообщение об их отсутствии и невозможности дальнейшей работы.

Если же питание и синхронизирующие сигналы поданы на платный соединитель микропроцессора, то оператору выдается сообщение о нормальном продолжении процесса эмуляции. После его получения оператор должен осуществить распределение памяти и портов ввода-вывода в образованной микроЭВМ и ВСЭ системе, установить желательную для него систему счисления, после чего он может приступить к вводу отлаживаемой программы с диска или ее прямому набору с клавиатуры.

Распределение памяти осуществляется путем задания принадлежности некоторой квантованной области встроенной памяти эмулируемой микроЭВМ или внутренней памяти ВСЭ. При этом физическая адресация внутренней памяти ВСЭ может не совпадать с присвоенной этой области памяти адресацией в образованной системе. Необходимо отметить, что некоторые области памяти ВСЭ, которые используются для хранения системных программ, являются запрещенными для пользователя, поэтому если оператор желает назначить эту область памяти во внутренней памяти ВСЭ, то он должен воспользоваться возможностью переадресации областей памяти и назначить эту зону в одной из свободных областей внутреннего ЗУ. Эти запрещенные области памяти, как правило, совпадают с областью памяти, в которой расположена точка адресации первой команды микропроцессора после его установки в исходное состояние.

Распределение портов ввода-вывода осуществляется аналогично.

Запуск программы, введенной оператором в память, осуществляется одной из двух команд: командой пошагового исполнения или командой непрерывного выполнения.

При подготовке этих команд к запуску пользователь указывает адрес или метку начала программы. Он может не указывать начальный адрес, в этом случае в качестве начального адреса будет использован конечный адрес, по которому программа была остановлена во время предыдущей эмуляции, или адрес микропроцессора, который он автоматически формирует после установки в исходное состояние, если это первая после включения питания эмуляция.

Дальнейшее исполнение запущенной программы зависит от ряда дополнительных условий, которые пользователь указывает следом за адресом запуска программы.

В качестве условий останова процесса выполнения программы могут выступать:

адрес или метка программы, до которой оператор желает ее выполнить;

обращение к определенному адресу программы, заданному непосредственно или меткой с операцией, описанной в условии останова (чтение или запись, стековые операции чтения или записи);

обращение к заданному порту ввода-вывода с указанной операцией ввода-вывода.

В команде могут быть указаны одно или более (чаще до двух) условий останова.

Поведение системы ВСЭ – эмулируемой микроЭВМ после останова определяется следующими операторами команды. Если за условием останова следует оператор запрещения прерывания, то останов по условию, описанному в команде, не будет выполняться, хотя сигнал, по которому производится останов, будет вырабатываться. Таким образом, этот режим работы может использоваться для синхронизации внешних диагностических средств, например логического или сигнатурного анализаторов, осциллографа.

Если же в команде вслед за условием останова следует оператор, определяющий останов, то возможны следующие варианты дальнейшего поведения системы:

останов с выводом на дисплей содержимого внутренних регистров микропроцессора, текущего и продвинутого значений счетчика команд, содержимого указателя стека;

переход к выполнению указанной в операторе подпрограммы, расположенной по адресу, заданному аналогично описанному выше.

В команде может быть указано несколько операторов (как правило, до четырех), определяющих останов с выводом на устройство отображения или переход к подпрограмме.

Условия возобновления эмуляции описываются еще одной группой операторов команды, которые позволяют описать следующие продолжения:

возобновление по счетчику, когда эмуляция продолжается до указанного в операторе числа повторений условия останова;

автоматическое возобновление эмуляции после каждого останова;

возобновление эмуляции с точки ее запуска, т.е. организация цикла "начало программы – условие останова";

возобновление эмуляции по условию, т.е. в операторе задается некоторое значение, при совпадении которого с полученным в процессе выполнения программы производится возобновление эмуляции; одновременно может быть указано до четырех операторов, которые могут определять содержимое памяти, регистров, регистра флагов, состояние определенного вывода микропроцессора, портов ввода-вывода.

Команда пошагового выполнения отличается от команды автоматического выполнения эмуляции добавлением одного оператора, который определяет общее количество команд или команд определенного типа в шаге, т.е. команда пошагового выполнения может выполняться с остановом после окончания каждой команды или же заданного в описываемом операторе количества команд, при этом может быть задано либо просто количество команд, либо количество команд переходов, вызовов подпрограммы или возвратов.

Возобновление выполнения программы после ее останова может быть существенно с помощью команд:

продолжения эмуляции, которая осуществляет запуск выполнения программы с точки останова;

запуска эмуляции подпрограммы, которая заставляет систему перейти к выполнению указанной в команде подпрограммы.

Еще одной командой, полезной при диагностировании устройств или отладке программ, является команда задания диапазона памяти. С помощью этой команды можно установить несколько диапазонов памяти внутри всего возможного для данного микропроцессора массива памяти.

После выполнения этой команды запуск и работа программ пользователя возможны только внутри этих диапазонов. При выходе программы за заданный диапазон ее выполнение останавливается.

Форматы команд, используемые в управляющей программе, приведены в приложении.

Как видно из описанных выше возможностей ВСЭ, система из ВСЭ и проверяемой микроЭВМ с точки зрения аппаратной неисправности позволит:

произвести стимуляцию любой части эмулируемой микроЭВМ;

сравнить результат работы стимулируемой схемы с эталонным или предполагаемым автоматически или визуально путем сопоставления предполагаемых и полученных значений, отображенных на экране дисплея;

заменить подозреваемый узел эмулируемой микроЭВМ аналогичным узлом ВСЭ и проверить правильность функционирования такой системы, определив, таким образом, обоснованность подозрений в неправильной работе подозреваемого узла, при этом если после замены этого узла на аналогичный узел ВСЭ полученная схема работает правильно, то ясно, что подозреваемый узел работает неправильно;

определить правильность функционирования шин адреса и данных эмулируемой ЭВМ путем проведения операций записи, чтения, ввода, вывода и анализа результата этих операций;

провести тестирование отдельных частей эмулируемой микроЭВМ или всей микроЭВМ с помощью тестов, которые подготовлены и хранятся в памяти ВСЭ;

произвести проверку работоспособности массива ОЗУ или порта ввода-вывода путем записи константы в этот массив памяти или порт ввода-вывода и сравнить записанную информацию с записываемой константой; при несовпадении значений выводится информация об ошибке и приводятся записываемое и реально записанное значения, что позволяет без больших программных затрат проверять правильность функционирования ЗУ и портов ввода-вывода, хотя и не дает возможности выявлять все неисправности ЗУ, например, неисправность, связанную с короткими замыканиями адресных шин, приводящими к одновременной записи информации в две и более ячейки памяти;

зациклить участок программы без внесения в нее дополнительных команд перехода и просмотреть правильность его исполнения с помощью осциллографа или логического анализатора, синхронизируясь от сигнала на выходе компаратора блока внутрисхемной эмуляции, который распознает условие останова программы:

распознать характер неисправности по выводимому на дисплей сообщению об ошибке, в котором даются сведения о получении неправильного кода команды, ошибке контрольной суммы, т.е. несовпадении контрольной суммы введенной программы с эталоном, о превышении времени ожидания сигнала готовности, неправильном статусе микропроцессора и т.д.;

определить неправильность прохождения программы или работы некоторого узла схемы по различию между прогнозируемым и действительным количеством машинных тактов, которое потребовалось для работы данного участка программы или узла схемы;

определить наличие питания и синхронизирующих сигналов на разъеме микропроцессора и затем локализовать эту неисправность с помощью осциллографа.

Все перечисленные возможности, которые предоставляет ВСЭ пользователю, позволяют инженеру, который производит поиск неисправности, локализовать отказ с точностью до устройства в составе микроЭВМ. В ряде случаев локализация может быть приведена с точностью до отказавшей микросхемы. Однако такая точная локализация возможна лишь на поверхности. Как правило, локализация отказавшего компонента схемы производится с помощью внешнего диагностического оборудования типа логических или сигнатурных анализаторов или осциллографов. В данном случае ВСЭ используется для стимуляции подозреваемого узла и формирования синхронизирующего сигнала.

Процедура поиска неисправности с помощью ВСЭ выглядит следующим образом:

1. Соединение ВСЭ и проверяемой микроЭВМ в единую систему.

2. Включение питания ВСЭ и проверяемой микроЭВМ.

3. Запуск операционной системы ВСЭ.

4. Вызов программы управления внутрисхемным эмулятором.

5. Если программа не инициализируется и на дисплее

отображается ошибка инициализации, необходимо проверить наличие на соединителе микропроцессора проверяемой микроЭВМ потенциалов питания и синхронизирующих импульсов с помощью осциллографа. После этого устранить обнаруженную неисправность в схеме питания или синхронизации.

6. Если вызов программы управления произведен нормально, то оператор производит установку необходимых ему областей памяти из массивов памяти, принадлежащих эмулируемой микроЭВМ и ВСЭ. Распределение памяти осуществляется с учетом предполагаемой неисправности, наличия и адресации области памяти в эмулируемой микроЭВМ.

7. Распределение портов ввода-вывода, которое производится также с учетом наличия тех или иных портов ввода-вывода в эмулируемой микроЭВМ, их исправности и необходимости в данной работе.

8. Установка выбранной оператором системы счисления, в которой производится ввод и отображение цифровой информации.

9. Загрузка с диска или ЗУ ВСЭ необходимой тестовой программы.

10. Запуск программы в одном из двух возможных режимов: автоматическом или пошаговом.

11. Если исполнение программы автоматически остановилось с сообщением об ошибке, то необходимо произвести анализ характера ошибки, просмотреть, если требуется, предысторию останова и на основе полученных данных приблизительно определить отказавший узел. Одним из наиболее часто встречающихся признаков неисправности является переход программы в область памяти, которая не предназначена для ее работы. Аналогичные случаи происходят и с адресацией портов ввода-вывода. Поэтому необходимо достаточно строго подходить к распределению памяти и портов ввода-вывода, не заказывая ненужные при работе области памяти и порты ввода-вывода, что заметно упростит первичную локализацию неисправности.

12. Если программа не вышла на нормальное окончание или заданную точку останова в течение некоторого достаточного интервала времени, то это означает, что внутри программы образовался непредусмотренный цикл. В этом случае осуществляют прерывание работы программы от

кнопки прерывания, после чего необходимо просмотреть предысторию останова. Однако эта информация в данном случае часто ничего не дает, особенно когда цикл образовался не внутри выполняемой программы, а внутри некоторой другой программы, куда основная программа вышла в результате неисправности.

13. В случае, описанном в п. 11, можно приступить к локализации неисправности в подозреваемом узле, используя или основную тестовую программу, в которой каким-либо способом организуется циклическая стимуляция нужного узла, или новую тестовую программу, которая составляется с учетом характера предполагаемой неисправности и записывается оператором в память ВСЭ или эмулируемой микроЭВМ. Выбор памяти, в которую заносится тестовая программа, зависит от характера неисправности и распределения памяти между ОЗУ и ПЗУ эмулируемой микроЭВМ. Необходимо учитывать такие моменты, как возникновение тактов ожидания в случае записи программы в несуществующую для эмулируемой микроЭВМ область памяти, которая задается внутри ВСЭ. Можно воспользоваться наложением ЗУ ВСЭ на область ПЗУ эмулируемой микроЭВМ. В этом случае можно проверить правильность адресации ПЗУ и другие схемные узлы, обеспечивающие его работу. После запуска короткой циклической программы можно произвести точную локализацию отказавшей схемы с помощью осциллографа или логического анализатора.

14. Часть неисправностей с признаками по п. 11 и 12 не может быть даже приблизительно локализована по предварительной информации. В этом случае необходимо произвести запуск тестовой программы по частям или в пошаговом режиме, отслеживая ситуацию, во время которой произойдет нарушение порядка исполнения программы. После определения момента нарушения работы программы производится анализ состояния регистров микропроцессора, содержимого памяти, состояния портов ввода-вывода, на основании которого производится предварительное определение неисправности, поиск которой уже можно произвести аналогично п. 13.

15. Если не выполняется программа пользователя, которая представляет из себя "Монитор", расположенный в ПЗУ эмулируемой микроЭВМ, то первоначально необхо-

димо убедиться в правильности считывания программы из ПЗУ. Если программа считывается неправильно, то произвести локализацию неисправности с помощью короткой циклической тестовой программы, осуществляющей считывание информации из неисправной ячейки ПЗУ, и осциллографа. Если ПЗУ исправно, то одной из вероятных неисправностей является неисправность ОЗУ. Проверка этого предположения с помощью ВСЭ очень проста. Производится перераспределение памяти, и область ОЗУ, которая используется программой "Монитор", назначается внутри памяти ВСЭ. После запуска программы "Монитор" она должна начать работать правильно, и пользователю необходимо тем или иным способом локализовать неисправность в ОЗУ эмулируемой микроЭВМ. Поскольку основной функцией программы "Монитор" является установление диалогового режима работы, то в процессе его работы должны быть задействованы соответствующие порты ввода-вывода, локализацию неисправности в которых можно произвести с помощью соответствующих тестовых программ, стимулирующих работу этих портов ввода-вывода.

16. Если инициализированная тестовая или другая программа закончилась нормально, то приступают к продолжению проверки функционирования других узлов микроЭВМ с помощью тестовых программ.

Описанная здесь методика поиска неисправности ни в коем случае не претендует на полноту, поскольку реальных ситуаций бывает очень много, однако намеченный подход к поиску неисправности может позволить пользователю начать свой собственный анализ конкретного отказа.

Отладка программ с помощью ВСЭ, в общем-то, мало отличается от поиска неисправности в аппаратной части микроЭВМ. Для поиска ошибок могут применяться широко распространенные способы расстановки в программе контрольных точек, прогон программы по частям, контроль времени исполнения программы, анализ содержимого регистров микропроцессора и памяти.

Как видно из материала данной главы, ВСЭ действительно является мощным инструментом подготовки, редактирования, наладки программ и диагностирования аппаратной части микроЭВМ.

8.4. ПОСТРОЕНИЕ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ПОИСКА ОШИБОК НА БАЗЕ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

В последнее время большое внимание уделяется разработке и применению экспертных систем – интеллектуальных программных структур, позволяющих оперировать знаниями, отчужденными от их носителей – людей-экспертов, и вырабатывать на их основе новые знания, строить свои умозаключения, опираясь на известные правила. Для построения экспертной системы необходимы следующие типы данных:

1) факты, присущие данной предметной области, определенным образом классифицированные и связанные между собой;

2) правила, позволяющие получать новые соотношения между фактами. Чаще всего эти правила имеют следующий вид: "Если (условие), то (действие)";

3) управляющая структура, которая представляет собой алгоритм, задающий последовательность применения правил.

Часто при решении задач из области технической диагностики представляется целесообразным построение диагностической системы, работающей по принципу экспертных оценок. Рассмотрим общий случай. Пусть имеется объект диагностирования (ОД). Для тестирования этого объекта составлен набор тестов T: $\{T_1 T_2 ... T_n\}$. Каждый тест из этого набора проверяет подмножество Q_i^o ОД. Это может быть маршрут графа программы, если объектом диагностирования является программа, или путь в логической схеме, активизируемый входным набором соответствующего теста T_i. Выполнение теста T_i приводит к появлению реакции R_{ik}, одна из которых соответствует правильной работе подмножества Q_i^o, а остальные – неправильной. Подмножества Q_i^o, в свою очередь, разбиваются на подмножества Q_i^1 более низкого уровня, каждый из которых проверяется своим тестом. Таким образом, проверки продолжаются до достижения подмножества Q^m, соответствующего разрешающей способности диагностической системы (Q^m может быть оператором программы, блоком логической структуры, выходом микросхемы).

Задача диагностической системы в данном случае – найти ответы на вопросы: исправен ли заданный блок? исправна

ли система в целом? если она неисправна, в каком блоке ее содержится неисправность? Возможная структура экспертной системы показана на рис. 8.3. Базой данных БД, содержащей декларированные знания, является набор фактов, относящихся к объекту диагностирования. Это, в первую очередь, совокупность подмножеств Q_i^j, связанных между собой отношением включения. Каждое подмножество может быть описано следующим образом: Q_i^j:$\{T_i^j, R_{ik}^j\}$. Тесты и соответствующие им реакции связаны отношением "вызывает": "тестовый набор T вызывает реакцию R". Это утверждение может быть истинным или ложным. Среди реакций необходимо отличать R_{i0}^j, соответствующую исправному состоянию подмножества Q_i^j. В процедурную структуру предлагаемой системы входят следующие правила:

Если "T вызывает R_0" – истина, то Q – исправно, иначе – нет.

Если Q исправно и $Q = \prod_{i=1}^{n} Q_i$, то все Q_i исправны. В зависимости от того, какого типа неисправная реакция получена при выполнении теста T_i^j, определяется стратегия дальнейших проверок подмножества Q_k^j:

Если "T_i^j вызывает R_{ik}^j" – истина, то следующим проверяется подмножество Q_k^{j+1}.

Управляющая структура представляет собой алгоритм, определяющий последовательность задания вопросов, т.е. собственно стратегию тестирования.

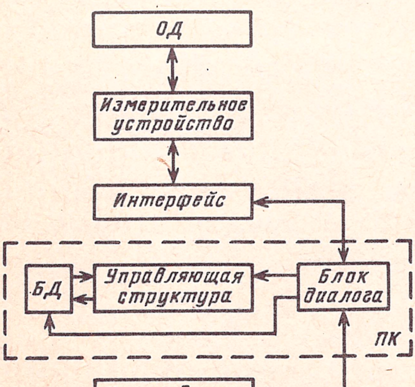


Рис. 8.3. Структура диагностической системы

Легко заметить, что в приведенную схему вполне укладывается конкретный алгоритм отладки микропроцессорных систем, предложенный в гл. 1. На основе работы этого алгоритма может быть построена экспертная система с базой данных, которая состоит из уровней отладки, выполняющих роль подмножеств Q_i, а каждый из уровней описывается функцией Ф. Сам же алгоритм выполняет роль управляющей структуры.

Очевидно, построенная на основе данного алгоритма экспертная система будет очень сложной за счет сложности базы данных и занимаемый объем памяти будет весьма велик. Рассмотрим часть системы, соответствующей аппаратному уровню логических сигналов в алгоритме на рис. 8.3. Для этого уровня база данных представлена следующими классами факторов:

1) команды ассемблера для диагностируемой МПС;

2) список аппаратных компонентов ОД;

3) списки точек наблюдения компонентов;

4) списки эталонных значений логических сигналов в этих точках.

Для определения управляющей структуры необходимо найти порядок выполнения проверок. Для этого множество подозреваемых компонентов, а также множество точек наблюдения каждого компонента должны быть упорядочены отношением влияния. Это отношение можно определить следующим образом: если ошибка в l_i проявляется в l_j, то l_i влияет на l_j. Естественно, что проверка должна начинаться с максимального элемента упорядоченного множества точек.

Для каждого компонента неисправность на линии управления влечет за собой неисправность на линии адреса или данных. Следовательно, проверка каждого компонента должна начинаться с проверки линии управления. Влияние же компонентов друг на друга в процессе выполнения команды может быть определено однозначно для каждой команды конкретного типа МП в результате анализа ее микропрограммы, временно́й диаграммы и т.д. Так, например, при выполнении команды MOV A, M (МП серии К580) прохождение потока данных определяется схемой на рис. 8.4. Это значит, что необходимо наблюдать точки и анализировать результаты в следующей последовательности:

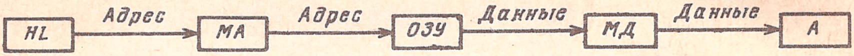


Рис. 8.4. Схема выполнения команды MOV A, M

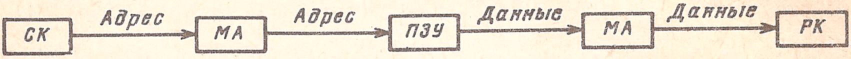


Рис. 8.5. Схема выборки команды в МП К580ИК80

$$HL_{МП}, L_{МА}, L^{A}_{ОЗУ}, L^{D}_{ОЗУ}, L_{MD}, A_{МП}.$$

Если не придерживаться этого порядка, то несоответствие вызова данных ОЗУ эталону укажет на ошибку в ОЗУ, тогда как, возможно, из-за неисправности в адресной магистрали ОЗУ выдает информацию из ячейки с адресом, отличным от того, что хранится в HL. Для фазы выборки любой команды схема прохождения потока данных показана на рис. 8.5 и порядок определяется так:

$$CK_{МП}, L_{МА}, L^{A}_{ПЗУ}, L^{D}_{ПЗУ}, L_{МА}, PK_{МП}.$$

В приведенном примере экспертной системы управления структура зависит от предметной области, т.е. от фактов, зафиксированных в базе данных, поскольку в зависимости от истинности того или иного факта сокращается и локализуется область дальнейшего поиска. При этом для каждого нового объекта диагностирования изменяется база данных, но процедурная структура (алгоритм локализации ошибок и правила, позволяющие делать вывод об исправности или неисправности) остаются неизменными.

Задание вопроса в диагностической системе представляет собой подачу тестового набора на входы ОД, а получение ответа на вопрос связано с наблюдением реакции. Поэтому в структуре экспертной системы присутствует средство наблюдения за ОД – измерительное устройство (ИУ). Поскольку задание вопросов – подача тестов – в различных подмножествах ОД может касаться различных параметров ОД, то измерительная часть системы должна быть в состоянии отражать ход выполнения программы ОД, индицировать в случае необходимости временные диаграммы или

таблицы истинности логических сигналов, производить измерение токов и напряжений. Представляется эффективным использовать в качестве измерительной части комбинированный цифровой диагностический прибор 43305.

При логическом анализе прибор 43305 позволяет определять неисправности с точностью до момента времени (команды, адреса и т.п.) и конкретного узла (например, разряда общей шины). При сигнатурном анализе может быть определена программно-временная область, в которой проявляется неисправность объекта. Бесконтактные измерения приращений напряжений и токов позволяют локализовать места обрывов печатных проводников. Токовый анализ локализует неисправности, характеризующиеся изменением проводимости в узлах печатных плат (короткое замыкание, пробой полупроводниковых приборов, одновременный выход на линию двух передатчиков и т.п.).

При логическом анализе в приборе 43305 отображение данных осуществляется на цифровом табло в виде 16-разрядной таблицы истинности в двоичном или шестнадцатеричном коде. Число каналов записи данных – 16, объем записываемых данных – 16 бит на канал, максимальная частота записи – не менее 5 МГц, минимальная продолжительность воспринимаемых логических состояний – не более 100 нс. Имеется четыре канала квалификатора. Обеспечиваются виды запуска – по одному условию и по последовательности двух–четырех условий.

Измерительный прибор является промежуточным звеном, осуществляющим передачу информации от объекта к системе, и наоборот, поэтому должны быть обеспечены программные и аппаратные интерфейсные средства между персональным компьютером и прибором 43305.

ПРИЛОЖЕНИЕ

СПИСОК КОМАНД ICE80 (КОМАНДЫ ПРИВОДЯТСЯ В АЛФАВИТНОМ ПОРЯДКЕ)

1. Команда BASE

Команда BASE предназначена для определения системы счисления или кода, в котором производится отображение информации (за исключением информации об адресах, по которым производится обращение к памяти или устройствам ввода-вывода, эта информация всегда выводится в шестнадцатеричной системе счисления).

Формат команды

BASE DECIMAL – задание десятичной системы счисления.

BASE OCTAL – задание восьмеричной системы счисления.

BASE HEXADECIMAL – задание шестнадцатеричной системы счисления.

BASE BINARY – задание двоичной системы счисления.

BASE ASCII – задание кода ASCII.

BASE SYMBOLIC – задание вывода данных в символьном виде.

2. Команда CALL

Команда CALL осуществляет передачу управления процессом эмуляции подпрограмме, начальный адрес которой задается в описании команды. После окончания выполнения подпрограммы эмуляция продолжается с точки, на которой она была прервана.

Формат команды

CALL ⟨адрес⟩,

где ⟨адрес⟩ – начальный адрес подпрограммы в шестнадцатеричной системе счисления или его символьное имя.

3. Команда CHANGE

Команда CHANGE предназначена для замены содержимого ячейки памяти, порта ввода-вывода, регистра и т.д. на информацию, задаваемую с клавиатуры.

Формат команды

CHANGE MEMORY ⟨адрес⟩ = ⟨значение⟩ { , ⟨значение⟩, ..., ⟨значение⟩} – осуществляется запись в заданную ячейку памяти с автоматическим увеличением адреса для записи следующего значения.

CHANGE DOUBLE MEMORY ⟨адрес⟩ = ⟨значение⟩ { , ⟨значение⟩, ..., ⟨значение⟩} – осуществляется запись двух байтов в две последовательно расположенные ячейки ОЗУ с соответствующим приращением адреса для записи следующего значения.

CHANGE REGISTER ⟨имя регистра⟩ = ⟨значение⟩ – запись в регистр.

CHANGE DOUBLE REGISTER ⟨имя регистровой пары⟩ = ⟨значение⟩ – запись в регистровую пару.

CHANGE FLAG ⟨имя флага⟩ = ⟨значение⟩ – замена содержимого флага.

CHANGE OUTPUT ⟨имя порта⟩ = ⟨значение⟩ – вывод в порт.

CHANGE VALUE ⟨символьное имя⟩ = ⟨значение⟩ – замена значения символа в таблице символов. Если символьное имя, заданное в команде, не существует в таблице символов, то генерируется сообщение об ошибке.

4. Команда DEFAULT

Команда DEFAULT предназначена для замены интервала поиска символьных ссылок. Если поиск символьной ссылки до использования команды DEFAULT производится в множестве всех модулей таблицы ICE80SD, то после ее выполнения поиск выполняется только в объявленных в команде модулях.

Формат команды

DEFAULT:SPEEN: – объявлен один модуль SPEEN.

DEFAULT:SPEEN:DEEP – объявлено два модуля SPEEN и DEEP.

5. Команда DISPLAY

Команда DISPLAY предназначена для отображения информации из источника, заданного в команде, на дисплее ВСЭ. Отображаемые данные выводятся в системе счисления, установленной в команде BASE.

Формат команды

DISPLAY REGISTER ⟨имя регистра⟩ – вывод содержимого регистра.

DISPLAY DOUBLE REGISTER ⟨имя регистровой пары⟩ – вывод содержимого регистровой пары.

DISPLAY FLAG ⟨имя флага⟩ – вывод содержимого флага.

DISPLAY PIN ⟨имя вывода⟩ – вывод состояния вывода микропроцессора.

DISPLAY MEMORY ⟨адрес⟩ – вывод из ячейки памяти.

DISPLAY DOUBLE MEMORY ⟨адрес⟩ – вывод из двух последовательных ячеек памяти.

DISPLAY INPUT ⟨имя порта⟩ – вывод из порта ввода-вывода.

DISPLAY VALUE ⟨символьное имя⟩ – вывод значения символа.

DISPLAY CYCLES ⟨количество отображаемых циклов⟩ – вывод информации о n последних машинных циклов, n не более 44.

DISPLAY STACK ⟨адрес⟩ – вывод информации из массива стековой памяти.

DISPLAY ALL REGISTERS – вывод содержимого всех регистров.

DISPLAY ALL PINS – вывод значения всех контактов микропроцессора.

DISPLAY ALL FLAGS – вывод содержимого всех флагов.

DISPLAY ALL SYMBOLS – вывод символьных имен и их значений.

DISPLAY ALL MODULES – отображение имен всех модулей.

DISPLAY NESTING – отображение начальных адресов подпрограмм.

6. Команда EQUATE

Команда EQUATE используется для добавления в таблицу символов нового символьного имени с соответствующим числовым значением.

Формат команды

EQUATE ⟨символьное имя⟩ = ⟨значение⟩.

Если вводимое символьное имя уже существует, то формируется сообщение об ошибке ERR = 6 и запись в таблицу символов не производится.

7. Команда EXIT

Команда EXIT осуществляет передачу управления программе "Монитор" или программе ISIS–II в зависимости от инициализации программы ICE80. Команда EXIT должна всегда инициализироваться перед выключением ВСЭ или проверяемого устройства или перед отключением кабеля, связывающего ВСЭ и микропроцессор проверяемого устройства.

Формат команды

EXIT

8. Команда FILL

Команда FILL предназначена для записи константы в заданный массив памяти. Границы массива и записываемая константа задаются в команде.

Формат команды

FILL MEMORY ⟨диапазон адресов⟩ WITH – ⟨значение⟩.

FILL DOUBLE MEMORY ⟨диапазон адресов ⟩ WITH ⟨значение⟩ ⟨диапазон адресов⟩ – ⟨начальный адрес⟩ TO ⟨конечный адрес⟩.

9. Команда GO

Команда GO инициализирует эмуляцию, т.е. осуществляет запуск программы, начальный адрес которой является одним из параметров команды. В команде GO определяются также условия останова эмуляции, действия, которые должен выполнить ВСЭ после останова, и порядок продолжения эмуляции с точки, в которой она была прервана.

Формат команды

GO FROM ⟨адрес⟩ UNTIL ⟨условия останова⟩ THEN ⟨действия во время останова⟩ CONTINUE ⟨условия продолжения эмуляции⟩ ⟨адрес⟩ – начальный адрес программы в шестнадцатеричной системе счисления или его символьное имя.

⟨условия останова⟩ – могут задаваться следующим образом:

⟨адрес⟩ EXECUTED – останов производится после выполнения команды, записанной по адресу, заданному в условии останова.

⟨адрес⟩ READ или WRITTEN – останов производится после считывания или записи информации по указанному адресу.

⟨адрес⟩ STACKED или UNSTACKED – останов производится после обращения с чтением или записью по указанному адресу, который должен адресоваться через счетчик стека.

⟨имя порта⟩ INPUT или OUTPUT – останов производится после ввода или вывода информации через указанный порт.

Если после условия останова указывается оператор NOBREAK, то останов не производится, но триггер совпадения условий останова срабатывает каждый раз, когда оно удовлетворяется. Выход триггера может использоваться для синхронизации измерительных приборов.

После выполнения останова ВСЭ выполняет действия, описанные после оператора THEN. ВСЭ может выполнить либо переход к подпрограмме MDS по оператору:

MDS CALL ⟨адрес⟩,

либо отобразить состояние регистров микропроцессора на дисплее, что выполняется по оператору:

DUMP.

Продолжение эмуляции после выполнения останова определяется условиями продолжения эмуляции, которые могут быть выбраны из следующих операторов:

CONTINUE COUNT ⟨*n*⟩ – выполнение прерванной программы возобновляется и продолжается до тех пор, пока условия останова не будут удовлетворены *n* раз. Останов производится при каждом совпадении условий останова.

CONTINUE FOREVER – возобновление эмуляции после каждого останова.

CONTINUE LOOPING – эмуляция возобновляется с точки ее запуска.

Регистры пользователя, регистры флагов, таймеры устанавливаются в исходное состояние, соответствующее моменту начала эмуляции по команде GO. Этот оператор может быть использован для синхронизации как и NOBREAK.

CONTINUE WHILE ⟨условия возобновления⟩ – эмуляция продолжается, если ⟨условия возобновления⟩ совпадают с реально полученным значением. ⟨Условия возобновления⟩могут содержать до четырех подчиненных условий, которые разделяются операторами END и OR, и записываются в виде:

⟨источник⟩ ⟨обозначение⟩ {MASK ⟨значение⟩} {⟨отношение⟩ ⟨значение⟩}, где ⟨источник⟩: MEMORY, или DOUBLE MEMORY, или REGISTER, или DOUBLE REGISTER, или FLAG, или PIN, или INPUT;

⟨обозначение⟩ – для памяти – ⟨адрес⟩, для остальных источников – их имя.

Содержимое ⟨источника⟩ поразрядно логически умножается на значение маски (или OFFFF (H), если маска опущена). Результат логического умножения сравнивается со значением по установленному отношению. В качестве отношения используются стандартные знаки математического отношения: =, =>, <=, >, <, < >. Если результат сравнения удовлетворяет указанному знаку, то эмуляция продолжается, если нет, то осуществляется останов.

10. Команда LIST

Команда LIST осуществляет назначение файла, в который производится вывод информации по командам DISPLAY, SEARCH, XFORM и по оператору DUMP при останове эмуляции. В качестве устройства, на которое осуществляется вывод, могут использоваться: дисковый накопитель, принтер и т.д.

Формат команды

LIST ⟨имя файла⟩.

Если в качестве устройства печати используется принтер, то команда имеет вид:

LIST:LP:

11. Команда LOAD

Команда LOAD предназначена для вызова таблиц символов, номеров операторов и абсолютных объектных кодов из специфицированных файлов. Объектные коды загружаются в физическую память, определенную командой XFORM. Таблицы символов и номера операторов запоминаются в специально выделенной в ВСЭ области памяти.

Формат команды

LOAD – при работе с MDS ICE80SD.

LOAD ⟨имя файла⟩ {NOCODE} {NOSYMBOLS} {NOLINES} – при работе с ISIS ICE80SD.

Управляющие слова NOCODE, NOSYMBOLS, NOLINES предназначены для запрета загрузки ненужных частей файла.

При вводе файла информация проверяется, и если обнаруживаются ошибки, то на дисплей выводятся сообщения.

12. Команда MOVE

Команда MOVE предназначена для пересылки содержимого одной области памяти в другую. Эти области памяти могут накладываться друг на друга.

Формат команды

MOVE MEMORY ⟨диапазон⟩ INTO MEMORY ⟨нач. адрес⟩;

MOVE MEMORY ⟨диапазон⟩ INTO MDS MEMORY ⟨нач. адрес⟩;

MOVE MDS MEMORY ⟨диапазон⟩ INTO MEMORY ⟨нач. адрес⟩;

MOVE MDS MEMORY ⟨диапазон⟩ INTO MDS MEMORY ⟨нач. адрес⟩;

⟨диапазон⟩ – ⟨нач. адрес⟩ TO ⟨конечн. адрес⟩;

⟨нач. адрес⟩ – адрес начала массива памяти, откуда или куда пересылается информация;

⟨конечн. адрес⟩ – адрес конца массива памяти, откуда пересылается информация.

13. Команда PAUSE

Команда PAUSE используется для передачи управления программе ''Монитор'' и может быть использована только при работе под управлением ISIS ICE80SD.

Формат команды

PAUSE

Возврат из ''Монитора'' осуществляется оператором G20.

13. Команда RANGE

Команда RANGE осуществляет установку диапазонов адресов памяти, которые используются в операторе THEN команд GO, STEP. При инициализации ВСЭ устанавливается диапазон адресов от 0 до 65535. Каждая последующая команда RANGE отменяет диапазоны, установленные ранее.

Формат команды

RANGE ⟨диапазон⟩ {, ⟨диапазон⟩ ... ⟨, диапазон⟩}.

14. Команда REMOVE

Команда REMOVE имеет три формата, которые служат для удаления соответственно одиночного символьного имени, всего модуля в целом и множества символьных имен.

Форматы команды

REMOVE ⟨символьное имя⟩.

REMOVE ⟨имя модуля⟩;

REMOVE ⟨символьное имя⟩ TO ⟨символьное имя ⟩.

Последний формат позволяет удалить все символьные имена, расположенные подряд между первым и вторым символьными именами, включая их самих.

15. Команда SAVE

Команда SAVE предназначена для записи содержимого указанной области памяти с таблицами символьных имен, таблицами номеров операторов (при работе с ISIS ICE80SD) и объектными кодами на дисковый накопитель или перфоратор (при работе в MDS ICE80SD).

Формат команды

SAVE ⟨диапазон⟩ – для MDS ICE80SD.

SAVE ⟨имя файла⟩ ⟨диапазон⟩ – для ISIS ICE80SD.

16. Команда SEARCH

Команда SEARCH предназначена для поиска заданного значения в определенном диапазоне адресов памяти. Все адреса ячеек памяти, в которых находится указанная информация, печатаются на дисплее ВСЭ.

Формат команды

SEARCH (DOUBLE) MEMORY ⟨диапазон⟩ MASK ⟨значение⟩ FOR ⟨значение⟩

⟨значение⟩, указанное в операторе MASK, логически умножается на содержимое считываемой ячейки памяти и сравнивается со ⟨значением⟩, указанным в FOR.

17. Команда STEP

Команда STEP осуществляет пошаговую инициализацию процесса эмуляции. Шаг задается специальным оператором.

Формат команды.

STEP {BY ⟨определитель шага⟩} FROM ⟨нач. адрес⟩ THEN ⟨действия во время останова⟩ CONTINUE ⟨условия продолжения эмуляции⟩, где BY определяет число шагов между остановами эмуляции, если оператор BY опущен, то останов выполняется после выполнения каждой команды.

⟨определитель шага⟩ имеет один из форматов:

⟨*n*⟩ INSTRUCTIONS – определяет размер шага в *n* команд.

⟨*n*⟩ JUMPS, или CALLS, или RETURNS – определяет размер шага в *n* команд JUMP (или CALL, или RETURNS).

⟨*n*⟩ – эквивалентно ⟨*n*⟩ INSTRUCTIONS.

Остальные операторы аналогичны операторам GO.

19. Команда TIMEOUT

Команда TIMEOUT служит для включения или выключения временного контроля выполнения команд при проведении эмуляции. При включенном временном контроле формируется сообщение об ошибке, если время выполнения какой-нибудь команды превышает 0,25 с.

Формат команды

TIMEUOT ENABLED – разрешение временного контроля.

TIMEOUT DISABLED – запрещение временного контроля.

20. Команда XFORM

Команда XFORM применяется для назначения областей памяти и портов ввода-вывода. Весь массив памяти ВСЭ разбит на 16 блоков по 4 Кбайта в каждом. Блоки нумеруются в десятичной системе счисления числами от 0 до 15. Мно-

жество портов ввода-вывода также разбито на 16 блоков по 16 портов в каждом, блоки портов ввода-вывода нумеруются аналогично.

Каждый блок памяти или порта ввода-вывода может быть назначен как в составе ВСЭ, так и в эмулируемом устройстве или запрещен к использованию. (На некоторые блоки наложены ограничения по их использованию.)

Формат команды

XFORM MEMORY ⟨диапазон⟩ {⟨новое состояние⟩};

XFORM IO ⟨диапазон⟩ {⟨новое состояние⟩}.

В данной команде операторы имеют значения:

⟨диапазон⟩ – определяет номер блока или ряд номеров блоков (например, 3 TO 5).

⟨новое состояние⟩:

GUARDED – запрещает использование указанных блоков. Любое обращение к ресурсам запрещенного блока вызывает сообщение об ошибке. При инициализации ICE80 все блоки запрещаются.

UNGUARDED – разрешает использование перечисленных блоков и размещает их логически и физически в проверяемой системе.

INTO ⟨значение⟩ – устанавливает взаимосвязь между логической и физической адресацией. Этот оператор разрешает использование блоков памяти или портов ввода-вывода, принадлежащих к проверяемой системе и указанных в операторе ⟨диапазон⟩. При этом эти блоки физически размещаются в блоках ВСЭ.

СПИСОК ЛИТЕРАТУРЫ

1. **Наумов Б.Н., Гиглавый А.В.** Микропроцессорная технология – основа перспективных ЭВМ массового применения// Микропроцессорные средства и системы. 1984. № 1. С. 7–11.

2. **Микропроцессоры:** Системы программирования и отладки/ В.А. Мясников, М.Б. Игнатьев, А.А. Кочкин, Ю.Е. Шейкин; Под ред. В.А. Мясникова, М.Б. Игнатьева. М.: Энергоатомиздат, 1985.

3. **Отладка** систем управляющих алгоритмов ЦВМ реального времени/ Под ред. В.В. Липаева. М.: Советское радио, 1974.

4. **Малиновский Б.Н., Слободянюк А.И., Погорелый С.Д.** Системное математическое обеспечение микроЭВМ и отладочной системы на базе микропроцессора К580ИК80// УСиМ. 1982. № 3. С. 30–34.

5. **Мильнер А.Д.** Инструментальный комплекс АРМ 2–05. Состояние и перспективы развития// Моделирование программных средств и архитектуры микропроцессорных устройств и систем. М.: МДИТП. 1984. С. 112–116.

6. **Бирман Г.** Техническое обслуживание в эксплуатации на подъеме// Электроника. 1985. № 10. С. 47–57.

7. **Каган Б.М., Мкртумян И.Б.** Основы эксплуатации ЭВМ. М.: Энергоатомиздат, 1983.

8. **Казьмина С.К.** Компактное тестирование// Автоматика и телемеханика. 1982. № 3. С. 173–187.

9. **Байда Н.П.** Контроль вычислительных устройств на этапе их изготовления// Киев: Общество "Знание" УССР. № 1. С. 58–67.

10. **Берштейн М.С., Романкевич А.М.** Метод статистического контроля логических схем// Кибернетика. 1974. № 1. С. 58–67.

11. **Единая** система электронных вычислительных машин. Операционная система. ТЕСТРАН. Руководство программиста/ Ц 51.804.001–01.Д11.

12. **Терминология** в технике микропроцессорных интегральных схем и микроЭВМ// Микроэлектроника и полупроводниковые приборы/ Под ред. А.А. Васенкова, Я.А. Федотова. М.: Советское радио. 1979. Вып. 4. С. 17–29.

13. **Микропроцессорные** комплекты интегральных схем. Состав и структура: Справочник/ Под ред. А.А. Васенкова, В.А. Шахнова. М.: Радио и связь, 1982.

14. **Гуляев В.А.** Организация систем диагностирования вычислительных машин. Киев: Наукова думка, 1978.

15. **Forbes R., Rutherford D., Steiglitz C.** A self-diagnosable computers// In: AFIPS Proc. Fall Joint Comp. Conf. 1965. Vol. 27. P. 1073–1100.

16. **Preparata F., Metze G., Chien R.** On the connection assignement of diagnosis system// IEEE Trans. Comp. 1967. V. 16, N 15. P. 848–854.

17. **Микросредства** управляющей вычислительной техники серии В7 (философия разработки)/ Н.Н. Шереметьевской, В.М. Долкарт, Г.Х. Новик и др.// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М.: 1980. Т. 62. С. 5–22.

18. **КТС ЛИУС.** Средства управления с переменной структурой (СУПС)/ Отраслевой каталог на серийно выпускаемое и перспективное оборудование. М.: 1980. Т. 4. Вып. 2. № 5.

19. **Кравченко В.С., Крамфус И.Р.** Основные направления развития средств проектирования и разработки микропроцессорных систем// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М.: 1986. Т. 80. С. 128–138.

20. **Columbia** 2000 data sheets// Columbia automatic test systems. Columbia Automation Ltd.

21. **Fluk** catalog 1982–1983// John Fluke Mfg. Co. Inc. 1982. P. 159–165.

22. **Mikrocomputer** development system achieves hardware. Software harmony/ Garrev, Hou, Lally, Walker// Electronics. May 29. 1975.

23. **Ситон Д.** Программные средства для автоматизации поиска неисправности в условиях производства и ремонта// Электроника. 1982. № 24. С. 43–50.

24. **Wray** Developing an M 6800 microprocessor Systems with the motorola exerciser// Electronics. September 9. 1976.

25. **Гордон, Надиг.** Локализация неисправностей в микропроцессорных системах при помощи шестнадцатеричных ключевых кодов// Электроника. 1977. № 5. С. 23–33.

26. **Анализаторы** логических состояний – новое слово в приборостроении// Электроника. 1976. № 5. С. 93–97.

27. **Богородицкий А.А., Пункевич В.С., Шумский Л.Д.** Логические анализаторы как средство наладки дискретной техники// Приборы и системы управления. 1981. № 10. С. 37–39.

28. **Струнски С.** Развитие методов технологического обслуживания аппаратуры на основе БИС// Электроника. 1977. № 22. С. 55–60.

29. **Бадагляччо Л., Каттерон Р.** Ускорение поиска ошибок в цифровых схемах// Электроника. 1977. № 33. С. 34–39.

30. **Кудряшов В.И.** Устройство сигнатурного анализа // Межотраслевой семинар "Контроль и диагностика сложных цифровых узлов на микросхемах". Тез. докл. М.: 1981. С. 22.

31. **Frohwerk R.A.** Signature analysis: A new digitalfield Service Method// Hewlett – Packard J. May 1977.

32. **Надежность** автоматизированных систем управления/ И.О. Атовмян, А.С. Вайрадян, Ю.П. Руднев и др. М.: Высшая школа, 1979.

33. **Deua Ph.** Pour la logique programmee come pour la logique cablee, la maintenance par analyse de signature // Electronique Industrielle. 1982. N 37/1–9–82. P. 57–59.

34. **Mayo E.** Multimeter Combines signature analizer with de Voltmeter and frequency counter// Electronics test. 1982. N 6. P. 84.

35. **Locator** 7201. Field logic tester// Solartron. 1980. June. P. 19.

36. **Joannou C.S.** Embedded software lets mC diagnose its own errors// EDN. 1981. V. 26. N 17. P. 208–214.

37. **Основы** технической диагностики/ В.В. Карибский, П.П. Пархоменко, Е.С. Согомонян, В.Ф. Халчев. М.: Энергия, 1976.

38. **Пархоменко П.П., Согомонян Е.С.** Основы технической диагностики. М.: Энергоиздат, 1981.

39. **Гуляев В.А.** Техническая диагностика управляющих систем. Киев: Наукова думка, 1983.

40. **Кузьмук В.В.** Сети Петри и моделирование параллельных процессов. Препринт-17, ИПМЭ АН УССР, Киев, 1985.

41. **Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.** Алгебра, языки, программирование. Киев: Наукова думка, 1975.

42. **Линьят Т.** Архитектура малых вычислительных систем. М.: Мир, 1981.

43. **Виленкин С.Я., Трахтенгерц Э.А.** Математическое обеспечение управляющих вычислительных машин. М.: Энергия, 1972.

44. **Мячев А.А.** Организация управляющих вычислительных комплексов. М.: Энергия, 1980.

45. **Гуляев В.А., Коростиль Ю.М.** Организация вычислительного процесса с функциями оперативного диагностирования в управляющих вычислительных комплексах. Препринт-405. Киев: ИЭД АН УССР, 1984.

46. **Харрари Ф.** Теория графов: Пер. с англ. М.: Мир, 1973.

47. **Форд Д., Фалкерсон Д.** Потоки в сетях: Пер. с англ. М.: Мир, 1966.

48. **Берж К.** Теория графов и ее применения: Пер. с англ. Изд-во иностр. лит., 1962.

49. **Минц Г.Е.** Теорема Эрбрана// Сб. статей "Математическая теория логического вывода". М.: Наука, 1977. С. 311–350.

50. **Кудряшов В.И.** Локализация отказов процессора микроЭВМ с помощью сигнатурного анализа// Всесоюзная науч.-техн. конф. "Повышение эффективности и качества продукции на базе развития распределенных систем управления и применения микропроцессорных устройств". М.: 1978. С. 60–62.

51. **Кирьянов К.Г., Соловейчик Э.Б.** К проектированию РЭА, ориентированной на диагностику сигнатурным анализом// Техника средств связи. Сер. Радиоизмерительная техника. 1980. Вып. 1. С. 9–84.

52. **А.с. 877547 СССР.** Устройство для диагностического контроля/ С.С. Жернова, В.И. Кудряшов, Р.Р. Пурэ, В.Н. Степанов// БИ. 1981. № 40.

53. **А.с. 1075247 СССР.** Устройство для захвата магистрали ЭВМ/ С.С. Жернова, В.И. Кудряшов, В.Н. Степанов// БИ. 1984. № 7.

54. **Питерсон У., Уэлдон Э.** Коды, исправляющие ошибки: Пер. с англ. М.: Мир. 1976.

55. **Chan A.Y.** Easy-to-use signature analyzer accurately troubles-shoots complex logic circuits// Hewlett–Packard J. 1977. N 5. P. 9–14.

56. **Кудряшов В.И.** Определение сигнатуры с помощью деления по mod 2// Межотраслевой семинар "Контроль и диагностика сложных цифровых узлов на микросхемах". Тез. докл. М., 1981.

57. **Кудряшов В.И., Творогов В.Б.** Автоматизация расчета сигнатур микропроцессорных устройств// Материалы семинара "Микропроцессоры в системах управления". Тез. докл. Пенза, 1985.

58. **Система** и аппаратура сигнатурного входного контроля интегральных микросхем при изготовлении управляющих вычислительных средств/ Л.С. Балкарей, И.Н. Блансевич, Г.Х. Новик, С.А. Шендерович// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М. 1980. Т. 62. С. 70–77.

59. **Новик Г.Х., Крамфус И.Р., Шендерович С.Д.** Язык формального описания двоичных векторов и программа вычисления их сигнатур// Автоматика и вычислительная техника. 1982. № 2. С. 44–47.

60. **Корн Г., Корн Т.** Справочник по математике для научных работников и инженеров. М.: Наука, 1978.

61. **Новик Г.Х.** О достоверности сигнатурного анализа// Автоматика и телемеханика. 1982. № 5. С. 157–559.

62. **Ван дер Варден Б.Л.** Алгебра. М.: Наука, 1976.

63. **А.с. 830391 СССР.** Устройство для функционально-параметрического контроля логических элементов/ Г.Х. Новик, В.В. Сташин, Е.И. Мазур, Ю.Г. Шибер.

64. **Ярмолик В.Н., Демиденко С.Н.** Генерирование и применение псевдослучайных сигналов в системах испытаний и контроля. Минск: Наука и техника, 1986.

65. **Многоканальный** сигнатурный анализатор для проверки многовыходовых цифровых устройств/ В.И. Засловский и др.// Вопросы радиоэлектроники. Сер. ТПО. Вып. 1. 1982. С. 114–119.

66. **Волков А.Ф., Ведешенков В.А., Зенкин В.Д.** Автоматический поиск неисправностей в ЦВМ. М.: Советское радио, 1969.

67. **Миронов Г.А.** Испытательные программы для контроля электронных цифровых машин. М.: Наука, 1964.

68. **Мультипроцессорные** системы и параллельные вычисления/ Под ред. Ф.Г. Энслоу. М.: Мир, 1976.

69. **Основы** технической диагностики/ Под ред. П.П. Пархоменко. М.: Энергия, 1976.

70. **Кудряшов В.И.** Использование сигнатурного анализа для диагностики микроУВМ ПМВ01 и межмашинной магистрали// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М. 1980. Т. 62. С. 78–83.

71. **Microprocessor** exerciser. Model 5001C// Hewlett–Packard. External stimulus for trubleshooting with signature analysis. 1980. June.

72. **Microprocessor** exerciser. Model 5001C// Hewlett–Packard. External stimulus for trubleshooting with signature analysis. 1980.

73. **Microprocessor** exerciser. Model 5001D// Hewlett–Packard, External stimulus for trubleshooting Z80 – Bazed systems with signature analysis. 1981. Sept.

74. **Маршал М.** Сигнатурный анализатор для смешанных логических схем// Электроника. 1980. № 25. С. 9–10.

75. **Сигнатурный** анализатор со стандартным интерфейсом// Электроника. 1982. № 14. С. 95.

76. **Кофман А.** Введение в прикладную комбинаторику. М.: Наука, 1975.

77. **Stefanski A.** Free-running signature analysis simplifies trubleshooting// EDN. 1979. February 5. P. 103–105.

78. **Маршал М.** Сигнатурный анализ приобретает все новых сторонников// Электроника. 1980. № 4. С. 96–98.

79. **Воителев А.И., Лукьянов Л.М.** Система сигнатурной проверки аналого-цифровых преобразователей// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М. 1980. Т. 62. С. 90–92.

80. **Nadig H.J.** Signature analysis, – concepts, examples and guidelines// Hewlett–Packard J. 1977. May. V. 28.

81. **Robach C., Saucier G.** Dynamic testing of control units// IEEE Trans. Comput. 1978. V. C–27, July. P. 103–105.

82. **Designing** for testability// Microprocessors and Microsystems. 1979. V. 3. N 1.

83. **Карпенко С.Л.** Сигнатурный контроль функционирования специального математического процессора ПСВ01// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М. 1980. Т. 62. С. 78–83.

84. **Центральные** устройства микросредств управляющей вычислительной техники серии В7/ В.М. Долкарт, В.Н. Степанов, И.С. Колтыпин и др.// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М. 1980. Т. 62. С. 28–36.

85. **Кудряшов В.И.** Поиск неисправности в микроЭВМ с помощью характеристических сигнатур// Межотраслевой семинар "Контроль и диагностика сложных цифровых узлов на микросхемах". Тез. докл. М. 1983. С. 45–47.

86. **Кирьянов К.Г., Соловейчик Э.Б.** К проектированию РЭА, ориентированной на диагностику сигнатурным анализом// Техника средств связи. Сер. Радиоизмерительная техника. 1980. Вып. 1. С. 9–84.

87. **A designers** guide to signature analysis// Application note 222, Hewlett–Packard. 1977. April.

88. **Sharrit D.** Team up a μP with signature analysis and ease trubleshooting in the field// Electronic design. 1979. N 1. January 4. P. 138–143.

89. **Signature** analysis test equipment// Electronic Engineering. 1982. December.

90. **Application** note 222–4. Guidelines for signature analysis. Understanding the signature measurement// Hewlett–Packard 1981. January. P. 34.

91. **Zachan M.P.** Hard, soft and transient errors in dynamic RAM// Electronics test. 1982. N 3. P. 42–54.

92. **Smith A.L.** Hard and soft failure in dynamic RAM fault tolerant memories// IEEE Trans. on reliability. 1981. V. R30. N 1. P. 58–60.

93. **Новик Г.Х.** Сигнатурный контроль интегральных микросхем оперативных запоминающих устройств методом перемещения информации (Тест "Марш")// Микроэлектроника. 1982. Т. 11. Вып. 1. С. 54–63.

94. **Жаркова П.Е., Савельев В.С., Сливицкий Ю.А.** Контроль полупроводниковых ЗУ// Электронная промышленность. 1975. № 8. С. 78–82.

95. **Кудряшов В.И., Новик Г.Х.** Сигнатурные анализаторы как средство формализации наладки и эксплуатации дискретной техники// Приборы и системы управления. 1984. № 3.

Глава 5. **Методы построения систем тестовых последовательностей сигнатурного анализа** 148

5.1. Способ формирования тестовых последовательностей сигнатурного анализа для стимуляции отдельных частей микропроцессорных устройств 148

5.2. Методика составления систем тестовых последовательностей для проверки отдельных устройств 156

5.3. Организация систем тестовых последовательностей сигнатурного анализа для МПК 161

Глава 6. **Разработка системы диагностирования микропроцессорного комплекса с использованием сигнатурного аналаза** 164

6.1. Устройство сигнатурного анализа для поиска неисправностей в МПК, реализованном на базе МСУВТ В7 164

6.2. Разработка центрального процессорного устройства с учетом требований диагностики и алгоритмы проверки устройств МСУВТ В7 167

6.3. Организация системы диагностирования комплекса цифрового управления 179

Глава 7. **Логические анализаторы и задачи отладки МПС** 182

7.1. Общие сведения о логических анализаторах как средстве диагностирования и отладки МПС 182

7.2. Способы запуска. Системы синхронизации 195

7.3. Применение логических анализаторов для отладки программного обеспечения 206

7.4. Последовательные алгоритмы комплексной отладки МПС 210

Глава 8. **Внутрисхемные эмуляторы как средство отладки микроУВК** 221

8.1. Общие сведения 221

8.2. Описание блок-схемы внутрисхемного эмулятора 225

8.3. Применение ВСЭ для диагностирования микроЭВМ 229

8.4. Построение интеллектуальной системы поиска ошибок на базе персонального компьютера 237

Приложение. Список команд ICE80 242

Список литературы 248

ОГЛАВЛЕНИЕ

Предисловие 3

Глава 1. **Общие вопросы отладки и диагностирования микропроцессорных систем** 5

1.1. Особенности микроУВК как объекта наладки и диагностирования . . . 5

1.2. Неисправности в цифровых устройствах 35

1.3. Общие вопросы отладки и диагностирования микропроцессорных систем 44

1.4. Организация процесса отладки и отладочные операции 50

1.5. Тестовое диагностирование узлов, блоков и устройств МПС 53

1.6. Методы проверки и отладки программного обеспечения 62

Глава 2. **Математические модели для описания процессов и объектов отладки в составе микроУВК** 63

2.1. Алгоритмические описания процесса отладки и элементарные диагностические операции 63

2.2. Математические модели для описания МПС и составления отладочных тестов и процедур 68

2.3. Модели вычислительного процесса 74

2.4. Вопросы рациональной организации процесса отладки и диагностирования 78

Глава 3. **Выбор структуры сигнатурных анализаторов, предназначенных для диагностирования микропроцессорных комплексов, и расчет основных параметров** 81

3.1. Выбор структуры сигнатурного анализатора для поиска неисправности в МПК 81

3.2. Разработка метода расчета сигнатур и алгоритмов их вычисления с помощью ЭВМ 91

3.3. Оценка достоверности сигнатурных анализаторов с различными генераторами псевдослучайных чисел 100

3.4. Многовходовые сигнатурные анализаторы и их применение 105

Глава 4. **Методы организации систем диагностирования МПК с использованием сигнатурного анализа** 118

4.1. Общие вопросы 118

4.2. Исследование методов и алгоритмов проверки МПК 124

4.3. Алгоритмы проверки устройств, входящих в состав МПК. Определение требований к составу дополнительного оборудования 131

4.4. Сокращение времени поиска неисправности при использовании метода сигнатурного анализа 141

96. **Средства** автоматизации управления электротехническими системами и изделиями// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М. 1980. Т. 62.

97. **Классификация** логических анализаторов/ П.П. Алексеев, Ю.Ф. Воленс, А.Ф. Тирфанов, В.Е. Добровинский// Вопросы радиоэлектроники. Сер. ЭВТ. 1984. Вып. 3. С. 13–18.

98. **Алексеев П.П.** Разработка и исследование алгоритмов функционирования логических анализаторов/ Вопросы радиоэлектроники. Сер. ЭВТ. 1984. Вып. 3. С. 13–18.

99. **Шлимович Е.М.** Логические анализаторы для проверки и наладки сложных цифровых устройств и систем// Вопросы радиоэлектроники. Сер. ЭВТ. 1980. Вып. 4. С. 27–42.

100. **Элиан Л.Е.** Логические анализаторы// Техника средств связи. Серия радиоизмерительная техника. 1981. Вып. 6. С. 1–22.

101. **Королев М.И., Мустафин Ф.Д., Хакназаров Х.Х.** Особенности использования логического анализатора ТАЛС 1604 для контроля и диагностирования современных средств микропроцессорной техники// Материалы семинара по разработке и внедрению систем контроля и диагностирования. РЭА. М. 1984. С. 83–91.

102. **Бирк Дж.** Личные системы, ускоряющие тестирование и снижающие его стоимость// Электроника. 1983. № 7. С. 27–36.

103. **Бойс Д.** Универсальный логический анализатор// Электроника. 1983. № 8. С. 26–31.

104. **Золло С.** Серия логических анализаторов фирмы GOULD// Электроника. 1985. № 24. С. 47–98.

105. **Ландау И.Я.** Применение ЦВМ для проектирования ЦВМ. М.: Энергия, 1974.

106. **Гуляев В.А., Макаров С.М., Новиков В.С.** Диагностика вычислительных машин. Киев: Техніка. 1981.

107. **Автоматизация** проектирования вычислительных систем. Языки, моделирование, базы данных: Пер. с англ. Мир, М.: Мир, 1979.

108. **Мартин Ф.** Моделирование на вычислительных машинах. М.: Советское радио, 1972.

109. **Городецкий В.Э., Тыдыков В.П.** Методы тестирования микропроцессорных схем// Зарубежная электроника. 1986. № 10. С. 20–34.

110. **КРАМ** – как средство для комплексной разработки и отладки программ/ А.К. Грабенко и др.// Тр. Всесоюз. науч.-исслед. ин-та электромеханики. М. 1986. Т. 80. С. 105–115.

111. **Hewlett–Packard** MEASUREMENT, COMPUTATION, SYSTEMS/ 1986.

Производственно-практическое издание

Гуляев Виталий Анатольевич
Кудряшов Владимир Иванович

АВТОМАТИЗАЦИЯ НАЛАДКИ
И ДИАГНОСТИРОВАНИЯ МИКРОУВК

Редактор издательства *З.И. Михеева*
Художественные редакторы: *А.А. Белоус, Т.А. Дворецкова*
Технический редактор *Е.В. Козлова*
Корректор *В.В. Кудряшова*

ИБ № 2367

Набор выполнен в издательстве. Подписано в печать с оригинала-макета 14.02.92. Формат 60 х 88 1/16. Бумага офсетная № 2. Печать офсетная. Усл. печ. л. 15,68. Усл. кр.-отт. 15,92. Уч.-изд. л. 15,34. Тираж 3000 экз. Заказ 886. С 074.

Энергоатомиздат, 113114, Москва, М-114, Шлюзовая наб., 10.

Отпечатано в Московской типографии № 9 НПО "Всесоюзная книжная палата" Министерства печати и информации Российской Федерации
109033, Москва, Волочаевская ул., 40.

36

·ЭНЕРГОАТОМИЗДАТ·

АВТОМАТИЗАЦИЯ НАЛАДКИ И ДИАГНОСТИРОВАНИЯ МИКРОУВК